Entwurf und Untersuchung eines Client-Server-Modells zum sicheren Markieren von Medien mit digitalen Wasserzeichen in Internetanwendungen

Diplomarbeit

für die Prüfung zum
Diplom-Ingenieur (Berufsakademie)

des Studiengangs Informationstechnik
an der Berufsakademie Mannheim
Gutachter Patrick Wolf

Vorgelegt von

Verena Henrich

Kurs TIT03ANS
September 2006

Ausbildungsstätte Fraunhofer IPSI Darmstadt
Gutachter Dr. Martin Steinebach

„Die Neugier steht immer an erster Stelle eines Problems, das gelöst werden will.“

Galileo Galilei

Kurzfassung

Digitale Wasserzeichen sind eine Sicherheitstechnologie, mit deren Hilfe beispielsweise Urheberrecht digitaler Medien (Audio, Video, Bild, etc.) nachgewiesen werden kann. Dabei werden den Medien nicht-wahrnehmbar Informationen hinzugefügt.

Wasserzeichenalgorithmen können sowohl „lokal“ als Anwenderprogramme als auch „remote“ als Web Services eingesetzt werden. Dabei stellt die lokale Variante ein erhöhtes Risiko für die Sicherheit der Wasserzeichenalgorithmen dar, da diese so beispielsweise durch Reverse-Engineering-Versuche angreifbar werden. Auf der anderen Seite müssen die zu markierenden Medien bei der „remote“-Variante zunächst zum Server transferiert werden, bevor sie dort markiert werden können. Da digitale Medien zum Teil erhebliche Datenmengen umfassen, stellen der Transport und dessen Sicherheit eine Herausforderung dar. Des Weiteren könnten Benutzer zögern, ihre Originalmedien zu einem fremden Server zu transferieren, da sie ihre Urheberrechte in Gefahr sehen.

Für das Markieren mit digitalen Wasserzeichen reichen meist bestimmte Teile der zu markierenden Medien aus, bei Audiodaten beispielsweise bestimmte Frequenzbänder und bei Videodaten etwa Teile einzelner Frames. Diese Arbeit schlägt eine Aufteilung des Markierungsprozesses vor: Es soll ein öffentlicher Teil auf Clientseite zur Medienanalyse und ein geschützter Teil auf der Seite des Servers zum sicheren Einbringen von Informationen dienen. Die Idee hinter dieser Aufteilung ist, dass sicherheitsirrelevante Funktionalität, wie etwa die Analyse zur Erkennung wesentlicher Medienteile, auf dem Client und nur sicherheitsrelevante Funktionalität auf dem Server ausgeführt wird.

Bei diesem Konzept muss die Sicherheit aus mehreren Blickwinkeln untersucht werden, um eine geeignete Kombination eines Client-Server-Modells zu erreichen. Auf der einen Seite muss die Sicherheit der Wasserzeichenalgorithmen gewährleistet sein, sodass diese nicht angegriffen werden können. Auf der anderen Seite muss auch die Sicherheit der zu markierenden Medien gewährleistet sein – Benutzer müssen sicher sein können, dass ihre Originalmedien nicht in unberechtigte Hände geraten könnten und so das Urheberrecht gefährdet wird. Am Ende dieser Arbeit steht eine Umsetzung des Client-Server-Modells auf Basis von Java-Servlets und -Applets.

Abstract

Digital watermarking is a security technology to proof e. g. ownership of digital media (audio, video, image, etc.) by adding unnoticeable further information to the media.

Watermarking algorithms can be used both local as application programs and remote as web services. In its local form watermarking algorithm are exposed to higher security risks since for example reverse-engineering-attempts become possible. On the other hand, the media must be transferred to the server in order to be marked remotely. As digital media often comprise a considerable amount of data, transport and its security pose a challenge. In addition, content owners could hesitate to transfer their original media to a server they do not control for security reasons.

In order to watermark digital media, it often suffices to watermark only parts of media, like certain spectral bands in audio or parts of individual frames in video. This work suggests a partition of the watermarking process: A public client-side component for analyzing the media and a private server-side component for secure embedding of information. The idea behind this partitioning is executing non-security-related functionality like identification of relevant parts of the media on the client side and only security-related functionality like the actual information embedding on server.

Utilizing this concept security needs to be analyzed in several perspectives for reaching a suitable combination of client-server-model. On the one hand, security of watermarking algorithms needs to be guaranteed so that they cannot be attacked. On the other hand, also security of the media needs to be guaranteed – content owners need to be sure that their original media are kept safe from unauthorized third parties. At the end of this work an implementation of such a client-server-model is presented that is based on Java-Servlets and -Applets.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich meine Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ort, Datum	Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Quelltextverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
AVI	Audio Video Interleave
BMP	Bitmap
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
GIF	Graphics Interchange Format
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
IPSec	Internet Protocol Security
JMS	Java Message Service
JPEG	Joint Photographic Experts Group
JVM	Java Virtual Machine
LSB	Least Significant Bit
MP2	MPEG-1 Audio Layer 2
MP3	MPEG-1 Audio Layer 3
MPEG	Moving Picture Experts Group
OSI	Open Systems Interconnection
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
UDDI	Universal Description, Discovery and Integration
URI	Uniform Ressource Identification
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAV	Waveform Audio Format
WSDL	Web Services Description Language
WWW	World Wide Web
XML	Extensible Markup Language
ZIP	Format zur komprimierten Archivierung

Kapitel 1

Einleitung

Der Einsatz digitaler Medien lässt vielfältige Möglichkeiten zu, bringt jedoch gleichzeitig die Problematik unbemerkter Vervielfältigungen mit sich. Die Notwendigkeit eines Client-Server-Systems zum sicheren Markieren von Medien mit digitalen Wasserzeichen wird im nächsten Abschnitt deutlich. Des Weiteren werden Schwierigkeiten und Anforderungen, die beim Entwurf eines solchen Systems auftreten und die es zu beachten gilt, angesprochen. Abschließend wird in diesem Kapitel die Struktur der Arbeit dargestellt.

1.1 Motivation

Die Verwendung von digitalen Medien ist in den letzten Jahren gewaltig gestiegen, dadurch werden analoge Medien nach und nach abgelöst [Dittmann00]. Digitale Medien können sehr einfach und beliebig oft ohne Qualitätsverluste vervielfältigt werden.

Zusammen mit der Entwicklung des Internets steigen die Möglichkeiten, Vervielfältigungen digitaler Medien zu verbreiten. Gleichzeitig wird die Gewährleistung von Urheberrechten erschwert, denn digitale Kopien weisen im Gegensatz zu Kopien analoger Medien keine Unterschiede zum Original auf. Zum Lösen dieser Problematik können digitale Wasserzeichen eingesetzt werden, die beispielsweise ein Urheberrecht durch einen Authentizitätsnachweis oder die Integrität von Daten nachweisen können. [Dittmann00]

Digitale Wasserzeichen bringen zusätzliche Informationen in digitale Medien ein. Sie sind untrennbar mit ihren Trägermedien verbunden und können beim Abspielen oder Betrachten von Menschen nicht wahrgenommen werden. Aktuelle digitale Wasserzeichenverfahren sind jedoch noch nicht so sicher, dass ihre Algorithmen freigegeben werden können, ohne dadurch effizientere Angriffe auf die Wasserzeichen zu ermöglichen [CaFoFu05]. Es muss demnach ein Szenario gefunden werden, bei dem digitale Medien mit digitalen Wasserzeichen markiert werden können, ohne dabei den Wasserzeichenalgorithmus offen legen zu müssen.

Da Medien einfach und schnell über das Internet verbreitet werden können, bietet es sich an, diese auch über das Internet mit digitalen Wasserzeichen zu markieren. Ein mögliches Szenario wäre ein Client-Server-Modell, bei dem sich der Wasserzeichenalgorithmus *sicher* auf dem Server befindet und die zu markierenden digitalen Medien vom Client zum Server transferiert werden, um dort mit digitalen Wasserzeichen markiert zu werden. Ein Problem hierbei ist die Größe des zu markierenden Mediums, die sich bei der Übertragung und dem Speicherbedarf auf dem Server bemerkbar macht. Bei einem kleinen Medium, beispielsweise einem Bild mit geringen Abmessungen, ist dies vielleicht nicht von Bedeutung, jedoch spätestens bei hoch aufgelösten Videos tritt dieses Problem verstärkt auf.

Ein weiteres Problem bei diesem Modell besteht darin, dass die Sicherheit der digitalen Medien ebenfalls gewährleistet sein muss. Das bedeutet, dass ein Benutzer auf Seite des Clients möglicherweise sein Originalmedium nicht ungeschützt zum Server transferieren möchte, denn dann besteht ein Sicherheitsrisiko während der Übertragung. Außerdem muss dem Server bzw. dem Betreiber des Servers soviel Vertrauen entgegengebracht werden, dass dort ein Original hingegeben wird.

Eine mögliche Lösung, bei der gleich beide Probleme behoben werden könnten, also das Problem der Mediengröße und das Problem der Sicherheit der digitalen Medien, besteht darin, dass clientseitig das digitale Medium analysiert wird und nur die für den Wasserzeichenalgorithmus zum Markieren notwendigen Teile des Mediums herausgesucht und zum Server übertragen werden. Das Problem der Mediengröße ist gelöst, denn wenn nur Teile des Mediums übertragen bzw. gespeichert werden müssen, dann ist die entstehende Mediengröße kleiner als die des Originals. Das Problem der Sicherheit der Medien ist offensichtlich auch umgangen, denn Teile eines Mediums stellen kein Originalmedium dar – somit ist das Vertrauensproblem aufgehoben.

Scheinbar sind diese beiden Probleme durch das vorgeschlagene Vorgehen gelöst. Die vorliegende Arbeit beschäftigt sich mit dem *Entwurf und der Untersuchung eines Client-Server-Modells zum sicheren Markieren von Medien mit digitalen Wasserzeichen in Internetanwendungen* unter Berücksichtigung der genannten Probleme. Dabei wird versucht, eine bestmögliche Lösung zu finden, bei der die Sicherheit aller beteiligten Systemteile sowie Medien berücksichtigt ist und auch die Mediengröße kein Problem darstellt.

1.2 Struktur der Arbeit

Um ein besseres Verständnis der vorliegenden Arbeit zu ermöglichen, werden in Kapitel 2 *grundlegende Begriffe* erläutert.

In Kapitel 3 werden die *Grundlagen* der für das Client-Server-Modell zum sicheren Markieren von Medien mit digitalen Wasserzeichen beteiligten Technologien geschaffen, mit dem sich diese Arbeit beschäftigt. Hierzu gehören digitale Medien, digitale Wasserzeichen, Client-Server-Architekturen und Sicherheitsaspekte.

Im vierten Kapitel wird eine *Untersuchung* der Voraussetzungen für das zu entwickelnde Client-Server-Modell angestellt. Dafür werden insbesondere die Interessen der Benutzer und Wasserzeichenentwickler herausgearbeitet und Wasserzeichenverfahren im Hinblick auf eine Aufteilung in Komponenten analysiert. Die allgemein herausgearbeitete Auf-

teilung von Wasserzeichenverfahren wird mit konkreten bestehenden Verfahren verglichen.

Im *Entwurf*, Kapitel 5, werden die Ergebnisse der Untersuchung konkretisiert. Dazu wird zunächst die Ausgangslage beschrieben und Bedingungen aus den Interessen der Benutzer und Wasserzeichenentwickler abgeleitet. Die Anforderungen an die Client- und an die Serverkomponente und ebenfalls an deren Zusammenspiel werden herausgearbeitet. Dabei werden die Ziele Sicherheit und Performanz besonders herausgestellt. Das Ergebnis dieses Kapitels ist schließlich ein Client-Server-Modell.

Dieses entworfene Client-Server-Modell mit den aufgestellten Anforderungen wird im sechsten Kapitel umgesetzt. Die *Implementierung* erfolgt zunächst soweit es geht allgemein, also nicht direkt an ein Wasserzeichenverfahren gebunden, bevor schließlich ein konkretes Wasserzeichenverfahren beispielhaft umgesetzt wird. In diesem Kapitel erfolgt außerdem die Diskussion der Umsetzung und der erreichten Ziele.

Abschließend folgt im siebten Kapitel eine *Zusammenfassung* der Arbeit mit einem Fazit der gewonnenen Erkenntnisse und einem Ausblick auf mögliche weiterführende Arbeiten zu diesem Thema.

Kapitel 2

Grundlegende Begriffe

Um eine einheitliche Basis der Begriffsdefinitionen für diese Arbeit zu bilden, sollen im Folgenden Erklärungen der für diese Arbeit relevanten Begriffe stattfinden. Dabei werden die Begriffe in Anlehnung an [Eckert05] definiert und verwendet.

2.1 Grundbegriffe

2.1.1 Aktive und passive Objekte

Dateien wie digitale Medien sind passive Objekte, sie können Informationen speichern und selbst verarbeitet werden. Aktive Objekte, wie beispielsweise Prozesse, können neben der Informationsspeicherung auch Informationen verarbeiten.

2.1.2 IT-System

Speicherung und Verarbeitung von Informationen findet in IT-Systemen statt. Solch ein System stellt ein Zusammenspiel technischer Komponenten dar, das Informationen speichern oder verarbeiten kann. Es kann über Schnittstellen von anderen Objekten des Systems oder von Subjekten bedient werden. Schnittstellen sind Operationen oder Methoden. Subjekte sind Benutzer oder Objekte, die im Auftrag von Benutzern aktiv sein können, wie Prozesse, Prozeduren und Server.

2.1.3 Zugriff

Eine Interaktion zwischen einem Subjekt und einem Objekt in einem System, durch die ein Informationsfluss zwischen Subjekt und Objekt auftritt, wird als ein Zugriff auf die Information bezeichnet. Jeder Zugriff auf ein Datenobjekt ist gleichzeitig auch ein Zugriff auf die dadurch repräsentierte Information. Für Zugriffe auf zu schützende Information bzw. auf die sie repräsentierenden Daten sind Zugriffsrechte festzulegen und an die Subjekte zu vergeben. ([Eckert05] Abschnitt 1.1)

In dieser Arbeit ist der Zugriff von Subjekten auf Informationen über vernetzte Systeme von Interesse. Vernetzte Systeme wie das Internet sind meistens durch Client-Server-Architekturen realisiert. Dabei treten entfernte Zugriffe von Subjekten wie Clients auf Server über Kommunikationsnetze auf.

2.1.4 Autorisierung

Besitzt ein Subjekt die Berechtigung zum Zugriff auf eine zu schützende Information bzw. auf ein zu schützendes Objekt, so ist das Subjekt zu diesem Zugriff autorisiert.

2.1.5 Transport

Findet ein Zugriff über ein Netz statt, so wird dieser Zugriff als entfernter Zugriff bezeichnet. Ein entfernter Zugriff erfolgt beispielsweise von einem Subjekt eines Client-Systems über ein Netzwerk auf ein anderes System, das Server-System. Durch das Netzwerk sind die beiden Systeme, Client- und Server-System, verbunden und somit stellen sie ein einziges System dar, ein Client-Server-System. Der Weg vom Anfang (zugreifendes Subjekt) bis zum Ende (Objekt auf das zugegriffen wird) eines Zugriffs, egal ob auf ein lokales oder entferntes Ziel, wird als Informationskanal bezeichnet. Wird ein Objekt von einem System in ein anderes übertragen, so spricht man von einem Transport über einen Informationskanal. In dieser Arbeit ist der Transport digitaler Medien zwischen Clients und Servern von Interesse.

2.1.6 Effizienz und Performanz

Die Effizienz beschreibt die Leistungsfähigkeit und Wirksamkeit eines Systems oder Algorithmus im Zusammenhang mit den Ressourcen Speicherplatz und Laufzeit. Sie bezieht sich auf das Verhältnis zwischen eingesetztem Aufwand und erreichtem Ergebnis. Die Auslastung der einzelnen Komponenten spielt dabei eine Rolle [Schneider91]. Mit steigender Effizienz sinkt die Komplexität eines Algorithmus. Ein ähnlicher Ausdruck wie Effizienz ist Performanz. Diese beschreibt ein konkretes Maß, wie beispielsweise die Datenrate, also den Durchsatz von verarbeiteten Daten pro Zeiteinheit.

2.2 Schutzziele

Schutzziele spezifizieren Zusicherungen über die Sicherheit eines Objektes oder Systems. Die Schutzziele Authentizität, Integrität und Vertraulichkeit sind bereits in der Kurzfassung und im einleitenden Abschnitt zu diesem Kapitel angesprochen worden. Jetzt sollen sie zusammen mit den beiden anderen in dieser Arbeit relevanten Schutzzielen Verbindlichkeit und Verfügbarkeit in Anlehnung an [Eckert05] Abschnitt 1.2 präzisiert werden.

2.2.1 Authentizität

Das Schutzziel Authentizität belegt die Echtheit und Glaubwürdigkeit eines Objekts bzw. Subjekts, indem es dieses eindeutig identifiziert und dessen Identität verifiziert. Um die Authentizität zu überprüfen, können charakteristische Eigenschaften mit der eindeutigen Identität verglichen werden. So kann die Authentizität eines Subjektes beispielsweise durch Passwörter oder digitale Signaturen (siehe Abschnitt 3.4.3) als charakteristische Eigenschaften nachgewiesen werden.

2.2.2 Vertraulichkeit

Die Vertraulichkeit von Informationen besagt, dass die Informationen nicht unautorisiert gewonnen werden können. Dafür müssen zuvor Berechtigungen festgelegt werden, welche Subjekte auf welche Informationen zugreifen dürfen bzw. welche Subjekte Kenntnis von welchen Informationen erlangen dürfen.

2.2.3 Integrität

Die Integrität schützenswerter Güter bezieht sich auf deren Veränderungen. Soll Integrität gewährleistet sein, darf keine unautorisierte und unbemerkte Veränderung der Daten stattfinden.

2.2.4 Verfügbarkeit

Die Forderung an ein System, authentifizierten und autorisierten Subjekten ihre Zugriffe gemäß ihrer Berechtigungen zu ermöglichen und gleichzeitig Subjekte in ihren Berechtigungen nicht durch unautorisierte Eingriffe zu beeinträchtigen, wird als Verfügbarkeit bezeichnet.

2.2.5 Verbindlichkeit

Ein System erfüllt die Verbindlichkeit bzw. Zuordenbarkeit von Aktionen, wenn nach deren Ausführung das auszuführende Subjekt eindeutig feststeht und es seine ausgeführten Aktionen nicht abstreiten kann. Ein bekanntes Beispiel zur Garantie der Verbindlichkeit, sind digitale Signaturen (siehe Abschnitt 3.4.3).

2.3 Sicherheit

Ganz allgemein kann Sicherheit die Abwesenheit von Gefahr bedeuten ([Dittmann00] Kapitel 2.2). Es können verschiedene Sicherheitsbegriffe unterschieden werden ([Eckert05] Abschnitt 1.1):

Funktionssicherheit (engl. safety): Ein funktionssicheres System kann keine unzulässigen Zustände annehmen.

Informationssicherheit (engl. security): Sie ist eine Erweiterung der Funktionssicherheit um die Eigenschaft, dass die Zustände, die ein System annehmen kann, keine unautorisierte Informationsgewinnung oder Veränderung zulassen.

Datensicherheit (engl. protection): Sie stellt ebenfalls eine Erweiterung der Funktionssicherheit dar. Die Datensicherheit ergänzt die Funktionssicherheit um die Eigenschaft, dass das System keine Zustände annehmen kann, bei denen unautorisierte Zugriffe möglich sind.

Datenschutz (engl. privacy): Dieser meint schließlich die Behandlung personenbezogener Daten nach Regeln des Bundesdatenschutzgesetzes.

Diese Sicherheitsbegriffe sind für die vorliegende Arbeit nicht alle gleich wichtig. Besonders der Datenschutz ist nur aus Vollständigkeitsgründen erwähnt, denn da die Behandlung personenbezogener Daten hier nicht betrachtet wird, ist dieser nicht von Interesse.

In den vorigen Abschnitten wurde der Begriff Sicherheit bereits in mehreren Zusammenhängen verwendet. Es wurde unter anderem über die Sicherheit digitaler Medien, die Sicherheit von Wasserzeichenalgorithmen oder –verfahren und über unsichere Netze gesprochen. Diese Aspekte der Sicherheit werden in den folgenden Abschnitten detaillierter besprochen.

2.3.1 Sicherheit digitaler Medien

Digitale Medien können einfach, schnell und ohne Qualitätsverluste vervielfältigt werden. Ebenso stellt ihre Verbreitung über das Internet kein Hindernis dar, sodass illegale Verbreitungen nicht ohne weiteres aufgespürt und nachvollzogen werden können. Ein Schutzziel digitaler Medien ist somit ganz eindeutig die Authentizität, denn ein Authentizitätsnachweis kann Urheberrecht sowie den Verbreiter illegaler Kopien nachweisen. ([Dittmann00] Abschnitt 2.2)

Digitale Medien können mit Hilfe moderner Programme leicht bearbeitet werden. Hier ist *bearbeiten* im Sinne von *verändern*, also *manipulieren*, gemeint. Die aufkommenden Fragen beziehen sich auf die Integrität und die Verbindlichkeit der Medien: Wie können die Schutzziele Integrität und Verbindlichkeit nachgewiesen werden, wenn digitale Medien so einfach zu bearbeiten sind? Woher weiß ein Benutzer, ob es sich um ein unversehrtes Medium handelt, bei dem die Integrität gewährleistet ist? Wie kann Fälschungssicherheit nachgewiesen werden, sodass auf digitale Medien Verlass ist? Eine Antwort auf diese Fragen können digitale Wasserzeichen sein (siehe Abschnitt 3.2).

2.3.2 Sicherheit des Wasserzeichenalgorithmus

Sicherheit des Wasserzeichenalgorithmus kann sich einerseits auf die *Sicherheit des Algorithmus als digitales Medium* beziehen. In diesem Fall gelten die gleichen Betrachtungen und Ergebnisse wie in Abschnitt 2.3.1. Andererseits kann mit der *Sicherheit des Wasserzeichenalgorithmus* auch die *Sicherheit des Wasserzeichenverfahrens* gemeint sein, sozusagen das Ziel digitale Medien untrennbar mit digitalen Wasserzeichen zu markieren, sodass die Wasserzeichen durch gezielte Manipulationen der Medien nicht zerstört werden können. Digitale Wasserzeichenverfahren sind in ihrer Entwicklung noch nicht so weit, dass sie dem Kerckhoffs-Prinzip entsprechen [CaFoFu05].

A. Kerckhoffs hat erkannt, dass es unrealistisch ist, einen Verschlüsselungsalgorithmus auf Dauer geheim zu halten [CaFoFu05]. Deshalb darf die Sicherheit eines Systems, das dem **Kerckhoffs-Prinzip** genügen soll, nicht von der Kenntnis der Ver- und Entschlüsselungsfunktion abhängen. Wenn die Algorithmen zur Ver- und Entschlüsselung frei zugänglich sind, dann darf der Schlüssel nicht berechenbar sein. Das Kerckhoffs-Prinzip besagt, dass die Sicherheit eines kryptografischen Verfahrens allein von den verwendeten Schlüsseln abhängen sollte. ([Eckert05] Kapitel 5.1)

2.3.3 Sicherheit von Client-Server-Systemen

Client-Server-Systeme sind zusammengesetzte, vernetzte Systeme, die aus einzelnen Komponenten bestehen, zu denen sowohl Clients und Server, als auch Netze für Zugriffe und Übertragungen zählen. Mehrere Clients können konkurrierend oder kooperierend auf Server zugreifen. Diese entfernten Zugriffe über unsichere Netzwerke sollen nur authentischen Clients erlaubt sein, wofür sich Clients und Server gegenseitig authentifizieren müssen.

Ein vernetztes System ist nicht komplett in sich abgeschlossen und nach außen abgeschottet, sondern es ist offen und verteilt. Es besteht aus mehreren Systemkomponenten, zu denen ein Netz zählt, das die anderen Komponenten verbindet. Die Sicherheit des Systems ist nur gewährleistet, wenn jede einzelne Systemkomponente sicher ist. Die Sicherheit kann also gefährdet sein, wenn beispielsweise Client oder Server unbemerkt unautorisiert betrieben werden oder wenn über das Netz die Integrität, Vertraulichkeit oder Verfügbarkeit angegriffen wird (siehe nächster Abschnitt). Ist nur eines der geforderten Schutzziele einer einzelnen Systemkomponente gefährdet, so ist die Sicherheit des gesamten Systems nicht gewährleistet.

2.3.4 Sicherheit von Objekten

Die Sicherheit von Objekten kann durch Schwachstellen und Angriffe beeinträchtigt werden. Schwachstellen können in Systemen durch Softwarefehler, unsichere Netze oder unsachgemäße Benutzung verursacht werden. Sie sind verwundbare Punkte, die Objekte für Bedrohungen anfällig machen. Bedrohungen nutzen Schwachstellen aus, um Systemen bzw. schützenswerten Gütern durch den Verlust der Verfügbarkeit, Vertraulichkeit, Authentizität oder Integrität Schaden zuzufügen.

Bedrohungen ergeben sich unter anderem aus Angriffen auf Systeme. Diese können nicht autorisierte Zugriffe oder Zugriffsversuche sein, wobei zwischen aktiven und passiven Angriffen zu unterscheiden ist. *Aktive Angriffe* richten sich gegen die Integrität von Objekten, sie modifizieren Objekte unautorisiert. *Passive Angriffe* hingegen richten sich gegen die Vertraulichkeit, also die unautorisierte Gewinnung von Informationen. ([Eckert05] Kapitel 1.3)

Kapitel 3

Grundlagen

Die fachlichen Grundlagen für die weitere Auseinandersetzung mit den Aspekten Sicherheit und Performanz im Zusammenhang mit dem Markieren von Medien mit digitalen Wasserzeichen in Internetanwendungen werden in diesem Kapitel gelegt. Der folgende Abschnitt beschreibt eine abstrakte Betrachtung digitaler Medien. Die darauf folgenden Abschnitte stellen digitale Wasserzeichen, Client-Server-Architekturen und Sicherheitsaspekte in vielerlei Hinsicht dar.

3.1 Digitale Medien

Mit dem Begriff *digitale Medien* sind alle Medien gemeint, die in digitaler Form gespeichert sind oder übertragen werden. Beispiele sind digitale Musikdateien (MP3, WAV), digitale Bilder (BMP, JPEG) oder Animationen (GIF), computergestützte Grafiken oder Filme (MPEG, AVI). Hier soll kein detaillierter Aufbau dieser unterschiedlichen Medienrepräsentationsmöglichkeiten gegeben werden, sondern es wird versucht, eine einigermaßen einheitliche, abstrakte Betrachtung zu entwickeln.

Abstrakt beschrieben bestehen alle digitalen Medien aus Bits, also Einsen und Nullen, welche Informationen darstellen, die je nach Interpretation unterschiedliche (Medien-) Repräsentationen ergeben. Diese bestimmen den Eindruck, den ein Betrachter bzw. Zuhörer erhält, und sind von physikalischen Modellen sowie von Wahrnehmungsmodellen abhängig. Digitale Medien dienen der Verbreitung und Darstellung von Information und können somit als digitale Trägermedien bezeichnet werden. Sie sind sehr einfach veränderbar und können beliebig oft ohne Qualitätsverluste vervielfältigt werden.

Digitale Medien sind (Daten-)Objekte, die Informationen speichern können. Die genaue Repräsentation der Information ergibt sich jeweils aus einer Interpretationsvorschrift. Es ist beabsichtigt, diese Informationen zu schützen, denn digitale Medien und deren Inhalte stellen schützenswerte Güter dar.

3.2 Digitale Wasserzeichen

Digitale Wasserzeichen beschreiben Verfahren, die es ermöglichen, versteckt Informationen in digitale Medien einzubetten [CoMiBl02]. Dabei stellen die digitalen Medien Trägermedien dar, denen zusätzliche Informationen (Wasserzeichennachricht) in Form von für den Menschen nicht-wahrnehmbaren Mustern hinzugefügt werden. Hierbei handelt es sich meist um Zufallsrauschsignale, in denen sich die Wasserzeichennachricht befindet ([Dittmann00] Kapitel 3).

Zur Erklärung der digitalen Wasserzeichen soll von den bekannten analogen Wasserzeichen ausgegangen werden. Diese werden beispielsweise bei Geldscheinen in Form des zu sehenden Bildchens verwendet. Wird in diesem Sinn der Ausdruck *Wasserzeichen* von analogen auf digitale Medien übertragen, so ist das eingebrachte Muster das digitale Wasserzeichen. Dieses Muster ist eigentlich nur eine andere Darstellung der eingebrachten Wasserzeichennachricht, wird jedoch häufig als digitales Wasserzeichen definiert ([Dittmann00] Kapitel 3). In der Literatur ist der Begriff digitales Wasserzeichen allerdings nicht immer eindeutig definiert. Damit eine präzise Aussage getroffen werden kann, muss zwischen dem Wasserzeichenmuster (Zufallsrauschsignal), das in das digitale Medium eingebracht wird, und der Wasserzeichennachricht, welche die einzubringende Information darstellt und durch das Muster repräsentiert wird, unterschieden werden [VoPePuEgSu01].

Abbildung 1: Markierungsprozess

Abbildung 2: Ausleseprozess

Das Verfahren digitaler Wasserzeichen ist sehr eng mit der Steganografie, dem Verstecken von Nachrichten in Trägermedien, verbunden. Beide Verfahren verwenden einen symmetrischen Schlüssel, das bedeutet, dass der Schlüssel des Markierungs- und

Ausleseprozesses identisch sind. Beim Markierungsprozess, siehe Abbildung 1, wird mit diesem und meist noch weiteren Parametern eine Wasserzeichennachricht in ein Originalmedium eingebracht.

Mit Hilfe des Ausleseprozesses hingegen, siehe Abbildung 2, wird die Wasserzeichennachricht aus dem markierten Medium ausgelesen. Dabei wird ebenfalls der geheime Schlüssel benötigt.

Ferner orientieren sich sowohl Steganografie als auch das Verfahren digitaler Wasserzeichen zum nicht-wahrnehmbaren und robusten Einbringen der Nachrichten gewisser Eigenschaften der Trägermedien beispielsweise an Wahrnehmungsmodellen. Der Unterschied zwischen beiden Vorgehensweisen besteht hauptsächlich in deren beabsichtigten Anwendungsziel [SiAt04]. In der Steganografie sollen Nachrichten geheim übermittelt werden, ohne dass deren Existenz überhaupt bekannt ist. Dafür werden Nachrichten unabhängig vom Trägermedium in dieses eingebracht, denn die Trägermedien dienen lediglich der Übermittlung von Informationen. Digitale Wasserzeichen beabsichtigen Nachrichten in Medien einzubringen, die in direktem Bezug zu den Trägermedien stehen, beispielsweise Nachrichten, die Informationen über die Medien selbst beinhalten. Hierbei ist die Existenz zusätzlicher Information meistens bekannt.

Die beiden Begriffe *nicht-wahrnehmbar* und *robust* wurden in diesem Abschnitt bereits ohne Erklärung verwendet. Sie sollen neben anderen im folgenden Abschnitt als Eigenschaften digitaler Wasserzeichen definiert werden. Abschnitt 3.2.2 beschreibt schließlich Anwendungsgebiete, in denen der Einsatz digitaler Wasserzeichen sinnvoll ist.

3.2.1 Eigenschaften

Digitale Wasserzeichen besitzen einige Eigenschaften, die je nach Anwendung unterschiedlich stark gewichtet sind. Zu den wichtigsten Eigenschaften zählen die Robustheit, die Transparenz, die Kapazität, die Performanz, die Komplexität und die Sicherheit.

Die **Robustheit** digitaler Wasserzeichen misst, in wie weit die eingebetteten Informationen auch nach Veränderungen des Trägermediums wieder ausgelesen werden können und kann als ein Spektrum von robust bis fragil beschrieben werden. Robust bedeutet in diesem Zusammenhang, dass die Wasserzeichennachricht auch nach Veränderungen des Trägermediums noch eindeutig auszulesen ist. Dabei umfassen Veränderungen sowohl übliche Verarbeitungsschritte als auch gezielte Manipulationen, die dazu dienen sollen, das Wasserzeichen zu zerstören. Dazu zählen beispielsweise Komprimierung, Skalierung, Konvertierungen in andere Datei- und Kompressionsformate sowie Digital-Analog- und Analog-Digital-Wandlungen. Damit digitale Wasserzeichen robust sein können, werden sie in essentielle Medienteile eingebracht, sodass sie robust gegenüber inhaltserhaltenden Veränderungen sind.

Das Gegenteil von robust ist fragil. Die Bezeichnung *fragiles Wasserzeichen* bedeutet, dass nach Veränderungen des Trägermediums das Auslesen der Wasserzeichennachricht nicht mehr möglich ist ([Dittmann00] Kapitel 3.1.3.1). Dabei können mehrere Fragilitäten unterschieden werden, bei denen das fragile Wasserzeichen nicht unbedingt bei der kleinsten Veränderung des Mediums zerstört wird, sondern gegenüber bestimmten Veränderungen robust ist ([Dittmann00] Kapitel 6). So können beispielsweise Inhalts-, Semi- oder selektive Fragilität differenziert werden.

Der Begriff *nicht-wahrnehmbares Wasserzeichen* bedeutet, dass das Wasserzeichen für den Menschen nicht-sichtbar (bei visuellen Medien) bzw. nicht-hörbar (bei akustischen Medien) ist. Im Zusammenhang mit digitalen Wasserzeichen wird statt *nicht-wahrnehmbar* meist der Ausdruck t*ransparent* verwendet. Die **Transparenz** gibt also an, wie stark ein Wasserzeichen wahrnehmbar ist. Sie kann als Maß für den Qualitätsverlust gesehen werden, der durch das Einbetten der Wasserzeichennachricht entsteht. Wasserzeichenverfahren mit hoher Transparenz zielen auf eine nicht-wahrnehmbare Manipulation, also einen geringen Qualitätsverlust des ursprünglichen Mediums.

Die Eigenschaft **Kapazität** digitaler Wasserzeichen beschreibt den möglichen Umfang (die Anzahl der Bits), den die einzubettende Wasserzeichennachricht annehmen kann. Um eine große Kapazität zu erzielen, muss das Medium eine ausreichend hohe Datenrate für die Wasserzeichennachricht bereitstellen. ([CoMiBl02] Kapitel 2.2.3)

Das Verhältnis zwischen der Dauer, die der Markierungsprozess digitaler Wasserzeichenverfahren benötigt, und der Abspieldauer des Trägermediums (Echtzeit) wird mit der Eigenschaft **Performanz** beschrieben. Dabei spielt die **Komplexität** des Wasserzeichenverfahrens eine Rolle, die den Aufwand beschreibt, der für das Einbetten und Auslesen des Wasserzeichens erbracht werden muss. Im Praxiseinsatz müssen der Markierungs- und der Auslesevorgang ausreichend schnell geschehen. Je nach Anwendung ist hierbei ein Vielfaches von Echtzeitgeschwindigkeit notwendig.

Die Eigenschaft **Sicherheit** digitaler Wasserzeichen wurde bereits in Abschnitt 2.3.2 angesprochen. Allgemein bedeutet hier Sicherheit, wie hoch der Schutz des Wasserzeichens vor gezielten Angriffen ist. Angriffe auf digitale Wasserzeichen werden in Abschnitt 3.4.5 vorgestellt. Hier genügt das Wissen, dass ein Angriff auf ein digitales Wasserzeichen eine gezielte Manipulation (im Gegensatz zu gewöhnlichen Veränderungen) des Mediums darstellt, mit der Absicht das Wasserzeichen zu zerstören ([CoMiBl02] Kapitel 9). Ein Wasserzeichen gilt als sicher, wenn man es ohne einen geheimen Schlüssel weder auslesen noch verändern kann, ohne dabei das Medium selbst zu zerstören – sogar dann, wenn man ein markiertes Medium besitzt und Kenntnis über das Wasserzeichenverfahren hat ([Dittmann00] Kapitel 3.1.3.4). Diese Beschreibung impliziert das Kerckhoffs-Prinzip.

Allen genannten Eigenschaften gleichzeitig maximal zu entsprechen ist für ein einziges digitales Wasserzeichenverfahren nicht sinnvoll und nicht möglich. Die Eigenschaften bzw. Anforderungsschwerpunkte konkurrieren miteinander und es hängt von der jeweiligen Anwendung ab, welche Eigenschaft wie stark gewichtet wird ([Dittmann00] Kapitel 3.1.3.8). Im nächsten Abschnitt werden einige Anwendungen vorgestellt, bei denen der Einsatz digitaler Wasserzeichen von Interesse ist.

3.2.2 Anwendungsgebiete

Zu den wohl bekanntesten Anwendungen digitaler Wasserzeichen gehört der *Nachweis der Urheberschaft* (proof of ownership) und die *Urheberidentifizierung* (owner identification). Dafür werden in die Trägermedien eindeutige Informationen über den Urheber eingebettet, wie beispielsweise Namen oder Personalnummern. Diese Anwendung erfüllt das Schutzziel der Authentizität und ebenfalls der Verbindlichkeit. Es kommen sehr robuste Wasserzeichen zum Einsatz, die auch nach Veränderungen des Mediums noch eindeutig auszulesen sind. ([Dittmann00] Kapitel 4)

Eine andere Anwendung, bei der ebenfalls Authentizität und Verbindlichkeit nachgewiesen werden, ist die *Kundenidentifizierung* (transaction tracking, traitor tracing). Dabei werden eindeutige Informationen über einen Kunden, wie etwa eine Kundennummer, mit einem robusten Wasserzeichen in die Medien eingebracht. Dadurch kann bei illegalen Kopien der unrechtmäßige Verbreiter zurückverfolgt werden. ([Dittmann00] Kapitel 5)

Das Schutzziel Integrität kann ebenso mit digitalen Wasserzeichen gewährleistet werden – es wird ein *Nachweis der Unversehrtheit* (content authentication) erbracht. Dabei muss zwischen binärer und semantischer Integrität unterschieden werden. Binäre Integrität bedeutet, dass das digitale Medium in keinem einzigen Bit verändert worden ist. Umgesetzt wird diese Forderung durch fragile Wasserzeichen, die bei der kleinsten Änderung des markierten Mediums zerstört werden. Die semantische Integrität erlaubt inhaltserhaltende Veränderungen am Medium und wird durch inhaltsfragile Wasserzeichen realisiert. Das bedeutet, dass ein Medium nur insofern verändert werden darf, dass seine Aussage erhalten bleibt. Es ist Definitionssache und vom Wasserzeichenverfahren abhängig, welche Veränderungen in welchem Maß zugelassen werden und welche nicht. Bei zugelassenen Veränderungen wird das Wasserzeichen nicht zerstört, es muss also robust gegen solche Veränderungen sein. Zugelassene Veränderungen sind beispielsweise Kompression oder Formatänderungen, bei denen die Aussage des Mediums nicht angegriffen bzw. verändert wird. Gegenüber unerlaubten Veränderungen ist das Wasserzeichen fragil, es wird zerstört, um die semantische Integrität zu gewährleisten. ([CoMiBl02] Kapitel 10)

Neben den genannten gibt es weitere Anwendungen, wie Kopierschutz (copy control), Annotationen oder Broadcast Monitoring, die jedoch für diese Arbeit nicht von Interesse sind. Sie können in [CoMiBl02] nachgelesen werden.

3.3 Client-Server-Architekturen

In diesem Kapitel wird in Abschnitt 3.3.1 zunächst der Aufbau verteilter Systeme erklärt. Dabei spielt die Kommunikation zwischen den beteiligten Komponenten eine große Rolle. Mögliche Kommunikationsformen sind Thema des Abschnitts 3.3.2.

3.3.1 Aufbau von verteilten Systemen

Erste Anwendungen in der Entwicklung von Rechensystemen, bei denen man von Clients und Servern sprechen kann, sind Print- und File-Server. Dabei gibt es in einem vernetzten System gleichberechtigte Rechner, die Clients, welche zentralisierte Dienste auf einzelnen Rechnern, den Servern, in Anspruch nehmen. [Bengel04]

Vernetzte Systeme mit Client-Server-Betrieb ermöglichen durch kooperative Verarbeitung eine Verteilung der Anwendung auf mehreren Rechnern im Netz. Dieser Zusammenschluss von Clients und Servern als Hardwarekomponenten über ein Netzwerk mit der logisch verteilten Anwendung wird als verteiltes System oder auch Client-Server-System bezeichnet. Dabei stellen die Komponenten Funktionseinheiten dar, die in einer Client-Server-Beziehung zueinander stehen und gemeinsam eine Funktion erfüllen, die eine einzelne Komponente allein nicht erbringen könnte. Ein Beispiel für ein weltweit bekanntes verteiltes System ist das World Wide Web (WWW). In diesem gibt es Web-Clients und Web-Server, die zur Kommunikation das Internet verwenden. [Bengel04]

Ein einfaches Client-Server-System besteht folglich aus zwei logischen Teilen: einem Server und einem Client. Der Server stellt dabei Daten oder Dienste zur Verfügung, die von dem Client verwendet werden können. Server und Client können auch mehrfach vorkommen. Sie stellen jeweils eigene Prozesse dar und besitzen verschiedene Zuständigkeitsbereiche. Die Interaktion zwischen ihnen ist fest vorgegebenen (siehe Abbildung 3) und kann wie folgt beschrieben werden: Ein Client sendet eine Anfrage (request) an einen Server. Dieser bearbeitet die Anfrage und sendet das Ergebnis als Rückantwort (reply) zurück an den Client. Dabei können Clients und Server auf einem oder auf verschiedenen Rechnern ablaufen, denn die entfernte Interaktion (remote), also der Zugriff auf einen anderen Rechner, soll sich dabei nicht von der lokalen Interaktion, bei der sich Client und Server auf einem Rechner befinden, unterscheiden. [Bengel04]

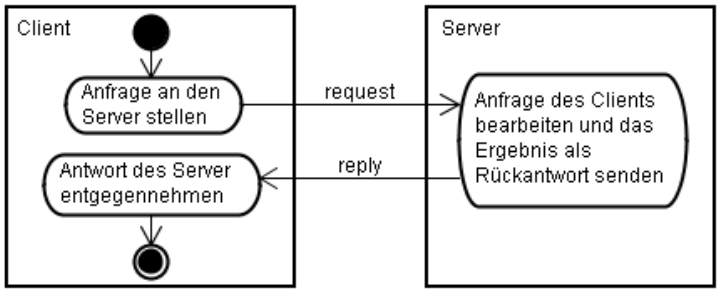

Abbildung 3: Interaktion zwischen Client und Server

Das beschriebene Prinzip der Interaktion gilt für Client-Server-Systeme im WWW. Dort spricht man von Hytertext Transfer Protocol (HTTP)-request und HTTP-reply.

Zur optimalen Zusammenarbeit der Prozesse muss zwischen Clients und Servern eine kooperative Verarbeitung stattfinden. Diese ist für ein Client-Server-System von zentraler Bedeutung und findet durch einen Informationsaustausch statt. Es können verschiedene Ansätze zur Kommunikation gewählt werden, die im nächsten Abschnitt vorgestellt werden.

3.3.2 Kommunikation und Koordination von Clients und Servern

Der Informationsaustausch als Kommunikation zwischen Client-, Server- und verteilten Prozessen muss koordiniert werden. Damit Kommunikation zwischen Rechnern stattfinden kann, muss zunächst eine Verbindung zwischen ihnen bestehen und sie müssen sich auf eine gemeinsame Kommunikationsform einigen. Dafür gibt es verschiedene Verfahren, wobei hier ausgewählte Beispiele vorgestellt werden.

Zu den einfachsten Verfahren zählt die **nachrichtenbasierte Kommunikation**, bei der die Kommunikation durch die Übertragung von Nachrichten erzielt wird ([Bengel04] Kapitel 3.1). Diese kann beispielsweise durch TCP/IP-Sockets oder Java Message Service (JMS) erfolgen.

Ein expliziter Nachrichtenaustausch zwischen Prozessen wird als **entfernter Prozeduraufruf** (Remote Procedure Call – RPC) bezeichnet. Dabei ist es Programmen erlaubt, auf Prozeduren zuzugreifen, die sich auf anderen Rechnern befinden. Als Information können vom Client Parameter übergeben werden, die der Server zur Ausführung des Aufrufs nutzt. Anschließend gibt der Server das Ergebnis zurück an den Client. ([TaSt03] Kapitel 2.2)

Als eine Erweiterung der RPCs kann die Kommunikation durch Aufrufe entfernter Objekte gesehen werden. Dabei kann der Client über Schnittstellen (Interface), die von jedem Objekt spezifiziert sind, Methoden aufrufen. Methoden, die sich auf einem entfernten Rechner befinden, können genauso aufgerufen werden wie lokale. Beispiele **objektbasierter Kommunikation** sind Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA) und Distributed Component Object Model (DCOM), deren genaue Funktionsweise in [Bengel04] Kapitel 3.3 nachgelesen werden kann.

In dieser Arbeit ist die **webbasierte Kommunikation** über das World Wide Web von Interesse. Das Web kann zahlreiche digitale Medien, die als Web-Objekte auf Web-Servern vorliegen, enthalten – sowohl Dokumente und Textinformationen als auch beliebige Daten wie Nachrichten, Datenbanken oder aber digitale Bilder, Videos und Audios. Interaktive Zugriffe von Web-Clients werden durch Anwendungsprogramme, Web-Browser, unterstützt. Dabei spielt der Uniform Resource Locator (URL) bzw. die Uniform Ressource Identification (URI) eine bedeutende Rolle. Diese stellen Referenzen auf Objekte dar und geben deren Speicherort auf dem Server an, damit Clients durch die Interaktion per HTTP auf sie zugreifen können. Spezielle Web-Objekte sind Web-Seiten, die in Hypertext Markup Language (HTML) geschrieben sind. Ihre Inhalte werden durch den Web-Browser interpretiert und angezeigt. Die einfachste Variante stellen dabei passive, statische Web-Seiten dar. Hypertext-Dokumente beinhalten häufig ausführbaren Code wie Java-Applets, der von Web-Clients heruntergeladen und ausgeführt werden kann. Solche Web-Seiten werden als aktive Dokumente bezeichnet. Die webbasierte Kommunikation unterstützt zudem dynamische Dokumente, bei denen der Server eine HTML-Seite aus serverseitigen Scriptsprachen dynamisch erzeugt, bevor sie an den Client geschickt wird. Beispiele zur Umsetzung dynamischer Web-Seiten sind das Common Gateway Interface (CGI) und Java-Servlets. Eine weitere Form der Nutzung webbasierter Kommunikation sind Web Services. Bei diesen handelt es sich um einen Standard des W3C (World Wide Web Consortium) [W3C06], der sich wiederum aus einer Reihe einzelner W3C-Standards zusammensetzt. Web Services werden in Abschnitt 5.6.2 genauer beschrieben. ([Bengel04] Kapitel 3.4)

3.4 Sicherheitsaspekte

Die Sicherheit eines verteilten Systems muss im gesamten System durchgängig realisiert sein, denn wenn eine Schwachstelle vorhanden ist, so wirkt sich die Unsicherheit auf das ganze System aus. Zur Gewährleistung eines sicheren Systems sind einerseits die Kommunikationssicherheit und andererseits die Authentifizierung der kommunizierenden Partner notwendig. Außerdem müssen die Schutzziele Zuverlässigkeit, Verfügbarkeit, Verbindlichkeit, Vertraulichkeit und Integrität gewährleistet sein. Dabei müssen Sicherheitsgefährdungen wie Lauschangriffe, Unterbrechungen oder Modifizierung verhindert werden. ([TaSt03] Kapitel 8)

Durch eine Sicherheitsstrategie wird festgelegt, welche Subjekte zu welchen Zugriffen bzw. Aktionen berechtigt sind. Sicherheitsstrategien werden durch Sicherheitsmechanismen erzwungen und sind beispielsweise Verschlüsselung, durch die Vertraulichkeit und Integrität gewährleistet wird, Authentifizierung zur Überprüfung der Identität von Subjekten, Autorisierung, um festzustellen, ob Subjekte entsprechende Berechtigungen haben, und Auditing, durch das Zugriffe verfolgt und Sicherheitsangriffe mit Gegenmaßnahmen beantwortet werden können. ([TaSt03] Kapitel 8.1)

In den folgenden Abschnitten werden einige Sicherheitsbetrachtungen angestellt, die sowohl Authentifikation, Zugriffskontrolle und kryptografische Verfahren, als auch die Kommunikation über Netze und deren Sicherheit beinhalten. Im letzten Abschnitt werden schließlich Angriffe auf digitale Wasserzeichen behandelt.

3.4.1 Authentifikation

Die Authentifikation realisiert das Schutzziel Authentizität eines Subjektes oder Objektes. Dafür muss eine eindeutige Charakterisierung durch definierte Eigenschaften vorgenommen werden, sodass über diese zweifelsfreien Merkmale eine eindeutige Identifizierung möglich ist. Die Authentifikation soll die Korrektheit einer behaupteten Identität kontrollieren, d. h. die Identität von Objekten bzw. Subjekten nachweisen. Dies kann durch mehrere Möglichkeiten realisiert werden, wie beispielsweise durch Wissen (z. B. Passwörter), persönlichem Besitz (z. B. Smartcards), digitale Zertifikate (siehe Abschnitt 3.4.3) oder biometrische Merkmale (z. B. Fingerabdrücke). ([Eckert05] Kapitel 7)

3.4.2 Zugriffskontrolle

Die Zugriffskontrolle hat die Aufgabe, Zugriffsversuche zu kontrollieren und Zugriffe nur autorisierten und berechtigten Subjekten zu ermöglichen. Zur Umsetzung der Zugriffskontrolle müssen vorher (Zugriffs-)Berechtigungen festgelegt werden, welche Subjekte auf welche Objekte zugreifen dürfen bzw. für Zugriffe autorisiert sind. Des Weiteren müssen die Subjekte authentifiziert sein, damit deren Autorisierung (Berechtigung) festgestellt werden kann.

3.4.3 Kryptografische Verfahren

Kryptografische Verfahren sind Methoden zur **Ver- und Entschlüsselung** von Daten. Generell können symmetrische und asymmetrische Verfahren klassifiziert werden, die sich in der Verwendung und der Anzahl der Schlüssel unterscheiden. Symmetrische Verfahren verwenden zum Ver- und Entschlüsseln den gleichen geheimen Schlüssel. Bei asymmetrische Verfahren hingegen wird zum Verschlüsseln der öffentliche Schlüssel des Empfängers und zum Entschlüsseln dessen geheimer Schlüssel verwendet. Mit diesen Verschlüsselungsverfahren können Informationen beispielsweise gegenüber Unbefugten (Angreifern) geheim gehalten werden – die Schutzziele Vertraulichkeit und Integrität der Daten werden realisiert.

Zum Nachweis der Authentizität, Integrität und Verbindlichkeit ist es nicht immer notwendig, Daten komplett zu verschlüsseln. Dafür reicht oft eine Codierung eines Hashwertes, der über die gesamten Daten berechnet worden ist. Diese Anwendung stellt eine **digitale Signatur** dar, bei der die Authentizität und Verbindlichkeit des Senders sowie die Integrität der Daten nachgewiesen werden können. Es wird zum Signieren ein privater Schlüssel und zum Entschlüsseln der dazugehörige öffentliche Schlüssel verwendet.

Um öffentlichen Schlüsseln vertrauen zu können, müssen diese beglaubigt sein. Diese Beglaubigung kann durch eine vertrauenswürdige Stelle (Zertifizierungsstelle) in Form eines **digitalen Zertifikates** erbracht werden. Ein Zertifikat enthält unter anderem Informationen über den öffentlichen Schlüssel des Inhabers und den Namen der Zertifizierungsstelle. Es ist in der Regel mit dem privaten Schlüssel der Zertifizierungsstelle signiert und kann somit mit deren öffentlichen Schlüssel überprüft werden. Die end-

gültige Entscheidung, ob ein Benutzer dem Herausgeber des Zertifikates vertraut, liegt jedoch bei diesem selbst.

Die genannten kryptografischen Verfahren, Verschlüsselung, digitale Signaturen und Zertifikate, können erst dann als sicher betrachtet werden, wenn sie dem Kerckhoffs-Prinzip entsprechen, da es unrealistisch ist, einen Algorithmus dauerhaft geheim halten zu können.

3.4.4 Sicherheit in Netzen

Um die Sicherheit in Netzen zu gewährleisten gibt es verschiedene Möglichkeiten, zu denen unter anderem Filter wie Firewalls gehören. Eine Firewall dient als eine Art Aufpasser am Eingang eines Netzes, die Datenpakete kontrolliert und filtert. Dabei werden nur die erwünschten (offenbar unkritischen) Pakete durchgelassen. Unerwünschte Pakete, die eine mögliche Bedrohung darstellen, werden hingegen am Durchgang gehindert. Auf diese Art und Weise wird hier eine Zugriffskontrolle vorgenommen.

Im Internet sowie auch in anderen Netzwerken spielt die Sicherheit der Kommunikation eine bedeutende Rolle. Eine Sicherung kann dabei in mehreren Schichten des Open Systems Interconnection (OSI)-Modells [ISO94], ein Schichtenmodell für die Kommunikation, ansetzen und es können verschiedene Verfahren zur Verschlüsselung (siehe voriger Abschnitt) und Authentifikation (siehe Abschnitt 3.4.1) eingesetzt werden. Zum Aufbau einer sicheren Verbindung sind Kommunikationsprotokolle wie Internet Protocol Security (IPSec) oder Secure Socket Layer (SSL) nutzbar.

SSL ist ein Protokoll, das auf der Transportebene des OSI-Modells aufsetzt und immer mit einem anderen Protokoll verwendet wird. Die Nutzung von HTTP über SSL wird als HTTP Secure (HTTPS) bezeichnet. Mittlerweile hat sich dieses im Internet für sichere HTTP-Verbindungen durchgesetzt, wobei vom Client über SSL eine sichere Verbindung zum Server aufgebaut wird. Über den so erzeugten SSL-Kanal können Daten über HTTP (oder auch andere Protokolle) ausgetauscht werden. ([Eckert05] Kapitel 9.4)

Der Einsatz von SSL gewährleistet eine vertrauliche Datenübertragung und die Integrität der transportierten Nachrichten. Dabei werden verschiedene Arten zur Authentifikation der Kommunikationspartner unterstützt, zu denen die wechselseitige, die einseitige und die anonyme Authentifikation gehören. Zur Authentifikation der Kommunikationspartner werden asymmetrische Verschlüsselungsverfahren und Zertifikate verwendet, wogegen bei der Übertragung der verschlüsselten Daten schließlich symmetrische Verschlüsselungsverfahren eingesetzt werden. ([Eckert05] Kapitel 9.4.)

3.4.5 Angriffe auf digitale Wasserzeichen

Angriffe auf digitale Wasserzeichen können nach mehreren Kriterien klassifiziert werden, beispielsweise nach ihrer Vorgehensweise, nach ihren Zielen oder nach der Kenntnis der Angreifer.

Grundsätzlich können zwei Vorgehensweisen bei Angriffen unterschieden werden. Bei den *indirekten Angriffen* entstehen Veränderungen, die beim normalen Umgang mit digitalen Medien auftreten. Hierzu können Größenänderungen, Kompressionsverfahren und andere Formatkonvertierungen gezählt werden, bei denen nicht unbedingt ein direkter Angriff auf das Wasserzeichen vorliegen muss. *Gezielte Angriffe* auf digitale Wasserzeichen sind beispielsweise das Einbringen von Rauschen oder Permutationen von Werten der Medien.

Neben ihrer Vorgehensweise können Angriffe nach ihren Zielen unterschieden werden. Dabei verfolgen die meisten das Ziel, das Wasserzeichen zu zerstören und gleichzeitig das Trägermedium zu erhalten [SiAt04]. Es gibt Angriffe, die auf eine komplette Entfernung abzielen (removal attack) [VoPePuEgSu01]. Bei anderen Angriffen wird versucht, die Synchronisation des Wasserzeichens für den Ausleseprozess zu verfälschen (geometric attack) oder weitere Wasserzeichen als Täuschung hinzuzufügen (cryptographic attack). Weitere mögliche Angriffe sind in [VoPePuEgSu01] nachzulesen.

Das Wissen des Angreifers über die für den Wasserzeichenalgorithmus relevanten Teile kann effizientere Angriffe auf das Wasserzeichenverfahren ermöglichen. Dies kann am Beispiel des Least Significant Bit (LSB-)Wasserzeichens verdeutlicht werden. Beim LSB-Wasserzeichenverfahren wird das Wasserzeichen in die *Least Significant Bits* eingebracht. Durch dieses Wissen ist es möglich, alle LSBs des markierten Mediums herauszukopieren, das Medium zu verändern und anschließend die zuvor herauskopierten LSBs wieder in das veränderte Medium einzufügen. Wird das Wasserzeichen aus den LSBs ausgelesen, so ist es trotz Veränderung des Mediums nicht zerstört, denn die LSBs sind die gleichen wie zuvor.

Neben der Klassifizierung der Angriffsziele und dem Vorgehen der Angreifer können auch die Kenntnisse der Angreifer unterschieden werden. Diese reichen von

1. *der Angreifer kennt nichts* über
2. *der Angreifer hat ein markiertes Medium* und
3. *der Angreifer kennt den Markierungsalgorithmus* bis zu
4. *der Angreifer kennt den Auslesealgorithmus* ([CoMiBl02] Kapitel 9).

Nach dieser Abstufung sind von 1. bis 4. immer effizientere Angriffe möglich, d. h., die Wahrscheinlichkeit, ihr Ziel zu erreichen, nimmt zu. Auf Grund dessen hat jedes Wasserzeichenverfahren bzw. jede Anwendung unterschiedliche Sicherheitsanforderungen ([CoMiBl02] Kapitel 9), um den 4 Fällen je nach Vorliegen gerecht zu werden.

Kapitel 4

Untersuchung

Ein Aufgabenteil dieser Arbeit ist die *Untersuchung eines Client-Server-Modells zum sicheren Markieren von Medien mit digitalen Wasserzeichen in Internetanwendungen*. Es liegt nahe, dass die Aufgaben für das Wasserzeichenverfahren auf Client und Server aufgeteilt werden, so dass eine höhere Sicherheit erzielt werden kann. Die Interessen von Benutzern und Wasserzeichenentwicklern sowie eine geeignete Aufteilung von Wasserzeichenverfahren werden in diesem Kapitel untersucht.

4.1 Analyse

Bei der abstrakten Betrachtung eines Wasserzeichenverfahrens kann festgestellt werden, dass dieses ein Programm darstellt, das als Eingabe ein (normalerweise lokal vorhandenes) Medium und Parameter erhält. Wenn die Ausführung eines Wasserzeichenverfahrens nicht lokal, also das Medium nicht beim Wasserzeichenalgorithmus vorliegt, sondern die Ausführung remote in einem Client-Server-System stattfinden soll, so ergeben sich zwei unterschiedliche Richtungen für Lösungsansätze:

1. Das Wasserzeichenverfahren wird komplett auf dem Server ausgeführt. Dafür muss zunächst das Medium vom Client zum Server hochgeladen werden. Es treten die Probleme Performanz des Systems und Sicherheit der Originalmedien auf.
2. Das Wasserzeichenverfahren wird komplett auf dem Client ausgeführt. Dafür muss der Wasserzeichenalgorithmus dem Client zur Verfügung gestellt werden. Es tritt das Problem der Sicherheit des Wasserzeichenverfahrens auf, denn durch dieses Vorgehen werden effizientere Angriffe ermöglicht.

Zwischen diesen beiden Ansätzen kann eine Skala aufgespannt werden, die von *alles auf dem Server* bis *alles auf dem Client* reicht. Der nächste Schritt besteht darin, einen Punkt auf dieser Skala zu finden, bei dem die gegebenen Anforderungen bestmöglich umgesetzt sind. Dabei muss die Sicherheit des Algorithmus und der Originalmedien gewährleistet sein und es dürfen keine erheblichen Performanzeinbußen auftreten. Für eine korrekte Aufteilung des Wasserzeichenverfahrens müssen Interessen von Benutzern und Entwicklern beachtet und einbezogen werden.

4.2 Interessen von Benutzer und Wasserzeichenentwickler

Beim Zusammentreffen von mehr als einem Subjekt kommt es oft zu einem Interessenkonflikt, bei dem eine Kompromisslösung gefunden werden muss. Die Benutzer in einem Client-Server-System, wie es hier entwickelt wird, sind mit den Clients gleichzusetzen. Die Wasserzeichenentwickler dagegen sind auf der Serverseite unterzubringen, denn sie werden ihre Wasserzeichenalgorithmen von dort den Benutzern anbieten.

Die Initiative, ein Client-Server-System bereitzustellen, kommt von dem Interesse des flexiblen Einsatzes digitaler Wasserzeichen. Dabei haben Wasserzeichenentwickler die Absicht, ihre Wasserzeichen Kunden anzubieten. Sie möchten ihre Wasserzeichenverfahren kommerziell zur Finanzierung der Entwicklungsarbeiten nutzen und durch die Bereitstellung ihrer Wasserzeichenverfahren diese verbreiten. Die Intension der Kunden besteht in der Anwendung digitaler Wasserzeichen, wobei sie ihre Medien damit markieren und so kenntlich machen möchten.

Zu den gemeinsamen Interessen von Kunde und Entwickler zählt die bestmögliche Performanz. Das bedeutet, dass das gesamte System ausreichend schnell bei der Ausführung sein und nicht unnötig Bandbreite sowie Speicherplatz verschwenden soll.

Ein gemeinsames Interesse von Benutzer und Wasserzeichenentwickler ist die Sicherheit des gesamten Systems. Dabei muss die Sicherheit von Client und Server in einem verteilten System und die Sicherheit des Netzes bzw. der Kommunikation berücksichtigt werden.

Weiterhin hat der Wasserzeichenentwickler das Interesse, den Wasserzeichenalgorithmus bzw. das Wasserzeichenverfahren zu sichern. Dafür darf der Algorithmus nicht an den Client herausgegeben werden, sondern das Markieren der Medien muss auf dem Server stattfinden.

Ein weiteres Sicherheitsinteresse des Clients ist die Sicherheit seiner Originalmedien. Diese kann gewährleistet werden, wenn die Medien nicht komplett herausgegeben, also auf den Server hochgeladen werden.

Aus diesen Interessen werden später Bedingungen für den Entwurf eines Client-Server-Modells abgeleitet (siehe Abschnitt 5.2). Im nächsten Abschnitt wird zunächst eine Aufteilung von Wasserzeichenverfahren untersucht.

4.3 Allgemeine Aufteilung von Wasserzeichenverfahren

Wasserzeichenverfahren bestehen meist aus mehreren Komponenten, in die sie je nach Implementierung des Verfahrens unterschiedlich aufgeteilt werden können. Mögliche Komponenten des allgemeinen Markierungsprozesses sind unter anderem folgende:

1. Analyse des Trägermediums und Heraussuchen der zum Markieren relevanten Medienteile
2. Markierungsalgorithmus (Markieren der herausgesuchten Medienteile)
3. Zusammenfügen markierter Medienteile mit Originalmedium

Zusätzlich können je nach Implementierung des Wasserzeichenverfahrens beispielsweise Wahrnehmungsmodellkorrekturen, Maskierungsverfahren oder Formatwandlungen hinzukommen. Bei den zuvor gelisteten Komponenten muss bedacht werden, dass der Markierungsalgorithmus Parameter benötigt, zu denen unter anderem ein geheimer Schlüssel und die einzubringende Wasserzeichennachricht gehören. Dabei kann die Wasserzeichennachricht bei den meisten Implementierungen nicht als Text eingebracht werden, sondern muss vor dem Einbetten kodiert werden. Das Ergebnis des Markierungsprozesses ist schließlich ein markiertes Medium.

Für den Ausleseprozess können ähnliche Komponenten herausgearbeitet werden:

1. Analyse des Trägermediums und Heraussuchen der Medienteile, in denen das Wasserzeichen potentiell enthalten sein kann
2. Auslesealgorithmus (Auslesen der Nachricht aus den herausgesuchten Medienteilen)

Der Auslesealgorithmus benötigt analog zum Markierungsalgorithmus einen geheimen Schlüssel. Das Ergebnis beim Ausleseprozess ist schließlich die ausgelesene Wasserzeichennachricht, die gegebenenfalls dekodiert werden muss, damit sie als Text vorliegt.

Diese allgemeine Einteilung in Komponenten gilt nicht pauschal für alle Wasserzeichenverfahren, denn jedes ist anders implementiert und wird unterschiedlich ausgeführt. Jedoch soll sie im Verlauf dieser Arbeit Verwendung finden, da sie beim Entwurf eines Client-Server-Modells zum Markieren von Medien nützlich ist. Für jede Komponente eines konkreten Wasserzeichenverfahrens müssen andere Sicherheitsaspekte betrachtet werden, damit die gesamte Sicherheit des Wasserzeichenverfahrens gewährleistet ist.

Die hier aufgeführten Komponenten werden im nächsten Abschnitt mit bestehenden Wasserzeichenverfahren verglichen, um zu zeigen, wie die allgemeinen Komponenten im konkreten Fall aussehen können. Dabei werden zunächst nur die Markierungsprozesse untersucht.

4.4 Aufteilung von konkreten Wasserzeichenverfahren

Jedes Wasserzeichenverfahren ist individuell implementiert und wird anders ausgeführt. Daher kann keine pauschale allgemeine Aufteilung in Komponenten vorgenommen werden. Stattdessen müssen die Möglichkeiten zum Aufteilen des Markierungsprozesses jeweils für ein konkretes Wasserzeichenverfahren betrachtet werden, da bei jedem andere Medienteile markiert werden, die zuvor herausgesucht werden müssen. Somit sind einzelne Verfahren für die Absichten der vorliegenden Arbeit besser geeignet als andere.

4.4.1 Komponenten beim Skalenfaktoren-Audiowasserzeichen

Das Skalenfaktoren-Audiowasserzeichen[1] [Steinebach99] wird in die Skalenfaktoren von Moving Picture Experts Group (MPEG)-1 Audio Layer 2 (MP2) eingebracht. Ein MP2-Audiostrom besteht aus kurzen Abschnitten, den Frames. Diese stellen jeweils einen bestimmten Zeitabschnitt dar, der je nach Datei von unterschiedlicher Dauer sein kann. Ein Frame besitzt einen Header und die eigentlichen Audiodaten. Die hier verwendeten Skalenfaktoren sind Bestandteile der Audiodaten. Sie bestehen jeweils aus 6 aufeinander folgenden Bits und stellen insgesamt ungefähr drei bis zehn Prozent des gesamten Audiostroms dar [StZm04]. Der genaue Aufbau von MP2 ist im Standard [ISO93] definiert und kann dort nachgelesen werden.

Das hier vorgestellte Verfahren besteht insgesamt – Markierungs- und Ausleseprosess zusammen – aus vier Executables: *makelist.exe*, *wminstxt.exe*, *link.exe* und *wmdet.exe*. Diese Executables müssen einzeln aufgerufen werden. Ihre Funktionalitäten sind die Folgenden [Steinebach99]:

1. **Makelist.exe** wird durch den Aufruf *„makelist audiofile.mp2 audiofile.lst“* ausgeführt. Eine Textdatei mit dem Namen *bitsplus.txt* wird zum Interpretieren des MP2-Audios im gleichen Verzeichnis benötigt. Dabei ist *audiofile.mp2* eine originale MP2-Datei. Diese wird interpretiert und eine Liste von Skalenfaktoreninformationen daraus generiert: *audiofile.lst*[2].
2. **Wminstxt.exe** bekommt durch den Aufruf *„wminstxt audiofile.lst audiofilew1.lst 2 0 -1 -2 -1 -1 0 1 3 5 5 3 MESSAGE“* die Skalenfaktoren-Datei (*audiofile.lst*) genannt. Diese Datei wird anhand von drei vorgegebenen Mustern verändert bzw. markiert und dadurch eine neue Liste erzeugt (*audiofilew1.lst*), in der nur die veränderten Frames enthalten sind. Die veränderte Datei kann also deutlich kürzer als die ursprünglich übergebene Skalenfaktoren-Datei sein. Die Zahlenparameter (*2 0 -1 -2 -1 -1 0 1 3 5 5 3*) des genannten Aufrufs sind Muster, die den geheimen Schlüssel darstellen. Der Textparameter (*MESSAGE*) gibt schließlich die einzubettende Wasserzeichennachricht an. Das Ergebnis der Ausführung dieser Executable ist eine markierte Skalenfaktoren-Datei (*audiofilew1.lst*), die eine Liste mit markierten Skalenfaktoreninformationen enthält.
3. **Link.exe** verbindet mit dem Aufruf *„link audiofile.mp2 audiofilew1.lst audiofilew1.mp2“* die Originaldatei (*audiofile.mp2*) mit der markierten Skalenfaktoren-Datei (*audiofilew1.lst*). Aus der ursprünglichen MP2-Datei und der markierten Skalenfaktoren-Datei wird eine neue, markierte MP2-Datei erzeugt (*audiofilew1.mp2*).

4. **Wmdet.exe** kann das Wasserzeichen wieder auslesen. Dies fordert zunächst eine erneute Ausführung von Schritt 1 („*makelist audiofilew1.mp2 audiofilew1.lst*“) auf der markierten MP2-Datei (*audiofilew1.mp2*). Dadurch erhält man eine Skalenfaktoren-Datei (*audiofilew1.lst*), die eine Liste mit Skalenfaktoreninformationen von der markieren Datei enthält. Durch den Aufruf „*wmdet audiofilew1.lst 2 0 -1 -2 -1 -1 0 1 3 5*“ wird aus dieser Skalenfaktoren-Datei (*audiofile1.lst*) das Wasserzeichen ausgelesen. Dabei stellen die übergebenen Zahlenparameter unter anderem wieder die drei Mustervorgaben dar.

Die beschriebenen Executables können den allgemeinen Schritten aus Abschnitt 4.3 zugeordnet werden. Beim Markierungsprozess erfüllt dabei *makelist.exe* den Schritt *Analyse des Trägermediums und Heraussuchen der zum Markieren relevanten Medienteile*. Der Schritt *Markierungsalgorithmus* wird durch *wminstxt.exe* ausgeführt. Das *Zusammenfügen markierter Medienteile mit Originalmedium* übernimmt schließlich *link.exe*. Beim Ausleseprozess ist *makelist.exe* der Schritt *Analyse des Trägermediums und Heraussuchen der Medienteile, in denen das Wasserzeichen potentiell enthalten sein kann* zugeordnet. Der *Auslesealgorithmus* ist im *wmdet.exe* implementiert.

Das Skalenfaktoren-Audiowasserzeichen liegt bereits in Komponenten geteilt vor, was dazu führt, dass es sich für die Absichten der vorliegenden Arbeit eignet. Aus diesem Grund ist es detaillierter als die anderen Verfahren beschrieben und wird beispielhaft umgesetzt. Weitere Verfahren werden lediglich kurz angesprochen, um weitere mögliche Umsetzungen zu untersuchen.

4.4.2 Komponenten bei einem Videowasserzeichen

Bei dem hier vorgestellten Videowasserzeichen werden nur die Helligkeitswerte von I-Frames markiert. I-Frames sind vollständige Bilder eines MPEG-Videos. Zum Markieren muss das MPEG-Video analysiert und die Helligkeitswerte der I-Frames herausgesucht werden. Der in Abschnitt 4.3 aufgezeigte Schritt *Analyse des Trägermediums und Heraussuchen der zum Markieren relevanten Medienteile* ist wieder zu erkennen. Der nächste essentielle Schritt besteht im eigentlichen Markieren dieser herausgesuchten Medienteile – der *Markierungsalgorithmus* wird angewendet. Die markierten Bildwerte müssen schließlich wieder mit dem Originalvideo in Verbindung gebracht werden – dies entspricht dem Schritt *Zusammenfügen markierter Medienteile mit Originalmedium*. Am Ende liegt ein markiertes Video vor.

4.4.3 Komponenten beim invertierbaren Audiowasserzeichen

Ein Teil eines Audios einer Audio-CD, ein so genannter Frame, hat die Länge N und besteht aus Abtastwerten s_1 bis s_N. Diese Werte liegen beim hier beschriebenen invertierbaren Audiowasserzeichenverfahren als 16-Bit-Integer vor, von denen nur eine der 16 Wertigkeiten betrachtet wird. Die vom Verfahren betrachtete Wertigkeit, meistens mindestens die 8. oder 9. Bitebene, muss komplett herausgesucht werden. Der erste Schritt ist also die *Analyse des Trägermediums* (Audio) *und das Heraussuchen der zum Markieren relevanten Medienteile* (gewünschte Bitebene).

Die ermittelte Ebene wird in allen Abtastwerten s_1 bis s_N verlustfrei komprimiert, wobei in der Praxis meist zwischen 100 und 1000 Bits pro Frame reduziert werden können. Der *Markierungsalgorithmus* nutzt diesen gewonnenen Platz aus, um ihn mit Wasserzeicheninformationen aufzufüllen. Ist die Markierung erfolgt, müssen die komprimierten Audiodaten inklusive der eingebrachten Wasserzeicheninformationen in das originale Audio integriert werden. Nach diesem *Zusammenfügen markierter Medienteile mit dem Originalmedium* liegt eine markierte Audiodatei vor.

4.4.4 Komponenten beim Zhao-Koch-Bildwasserzeichen

Das Zhao-Koch-Bildwasserzeichenverfahren [ZaKo95] bettet die Wasserzeichennachricht in ausgewählte Bildteile, so genannte Blöcke, ein. Es besteht aus zwei Teilen: Zum einen werden pseudozufällig Blöcke ausgewählt und zum anderen wird in diese Blöcke das Wasserzeichen eingebettet.

Hierbei sind die Schritte *Analyse des Trägermediums*, *Heraussuchen der zum Markieren relevanten Medienteile* und *Markierungsalgorithmus* wieder zu finden. Nach der Aufteilung aus Abschnitt 4.3 fehlt jedoch das abschließende *Zusammenfügen markierter Medienteile mit Originalmedium*, denn hier ist das Heraussuchen nicht im Sinne von Separieren gemeint, sondern es werden einfach die Blöcke bestimmt, in die das Wasserzeichen eingebettet werden soll. Da die Markierung direkt am Medium und nicht an gesonderten Medienteilen vorgenommen wird, muss hinterher nichts wieder zusammengefügt werden.

Demzufolge besitzt dieses Wasserzeichenverfahren in seiner ursprünglichen Form einen Schritt weniger. Durch Anpassungsmaßnahmen, bei denen die zu markierenden Blöcke nicht nur bestimmt, sondern auch separiert werden und dann isoliert markiert werden, kann der Schritt des Zusammenfügens nachträglich hinzugefügt werden. Für die Umsetzung in dem hier entwickelten System wäre dieser Schritt notwendig. Dabei kommt jedoch die Frage auf, ob es überhaupt sinnvoll und nicht zuviel Aufwand ist, diese Komponente hinzuzufügen.

---

[1] Dieses Wasserzeichenverfahren ist in [Steinebach99] entwickelt worden. Da dort jedoch kein Name genannt wurde, wird in dieser Arbeit der Ausdruck *Skalenfaktoren-Audiowasserzeichen* als Bezeichnung für dieses Wasserzeichenverfahren verwendet.

[2] Zur Vereinfachung wird eine Datei, die eine Liste von Skalenfaktoreninformationen enthält und mit *.lst endet, im Folgenden auch als *Skalenfaktoren-Datei* bezeichnet.

Kapitel 5

Entwurf eines Client-Server-Modells zum Markieren von Medien

Im vorigen Kapitel wurden die Grundlagen des Entwurfs analysiert – sowohl die Aufteilung von Wasserzeichenverfahren in Komponenten, als auch die Interessen von Benutzern und Wasserzeichenentwicklern. In diesem Kapitel soll ein Gesamtkonzept der aufgeteilten Markierung in einem Client-Server-System, unter Berücksichtigung der Interessen von Benutzer und Wasserzeichenentwickler, entworfen werden. Für den Entwurf des Modells wird zunächst mit der Beschreibung der Voraussetzungen und den Bedingungen begonnen. Danach werden die Aufgaben von Client und Server dargelegt und in Abschnitt 5.4 als konkretes Verfahren das Skalenfaktoren-Audiowasserzeichenverfahren in das entworfene Modell integriert. Des Weiteren werden die Möglichkeiten der Umsetzung von Client- und Serverkomponente und das Zusammenspiel dieser Komponenten beschrieben.

5.1 Voraussetzungen

Das Ergebnis des Entwurfs soll ein Client-Server-Modell sein, bei dem das Markieren digitaler Medien möglich ist. Client-Server-Modell bedeutet generell, dass ein Client und ein Server zusammenarbeiten, um eine gemeinsame Aufgabe zu erfüllen. Die Details der Zusammenarbeit werden in diesem Kapitel herausgearbeitet. Dabei soll das Modell möglichst die Interessen der Benutzer und Wasserzeichenentwickler beachten, wofür Bedingungen festgelegt werden müssen, die beim Entwurf dieses Modells eingehalten werden müssen. Die Bedingungen berücksichtigen die Interessen von Benutzer und Wasserzeichenentwickler und werden im nächsten Abschnitt aufgestellt.

Aus Abschnitt 4.1 der Untersuchung geht hervor, dass es beim Markieren mit digitalen Wasserzeichen in einem Client-Server-Modell zwei unterschiedliche Möglichkeiten der Aufgabenverteilung zwischen Server und Client gibt. Dies geht von *alles auf dem Server* bis *alles auf dem Client*, wobei ein geeigneter Punkt zwischen *den beiden Grenzwerten* gefunden werden muss. Hierbei ist zu beachten, dass alles, was auf dem Client ausgeführt wird, Reverse-Engineering-Angriffen ausgesetzt sein kann. Das bedeutet, dass ver-

sucht wird, kompilierten Code wieder in seine ursprüngliche Quelltextform zu bringen. Folglich kann alles, was sich auf dem Client befindet, als öffentlich gesehen werden. Diese Tatsache ist besonders für die Wasserzeichen relevant, wenn die Effizienz von Angriffen auf digitale Wasserzeichen nach den Kenntnissen der Angreifer aus Abschnitt 3.4.5 abgestuft wird.

Bei der Betrachtung der Sicherheit eines Systems, bei dem mehrere Subjekte miteinander interagieren, besteht oft ein Interessenkonflikt zwischen den Kommunikationspartnern, denn sie verfolgen meist unterschiedliche Schutzziele. Es muss demnach ein Kompromiss zwischen den einzelnen Sicherheitsinteressen der Kommunikationspartner gefunden werden.

In diesem Kapitel wird herausgearbeitet, welche Komponenten des Wasserzeichenverfahrens, wo – auf dem Client oder auf dem Server – ausgeführt werden sollten. Dabei werden die Interessen der Benutzer und der Wasserzeichenentwickler berücksichtigt, die im nächsten Abschnitt als Bedingungen an das Modell aufgestellt werden.

5.2 Bedingungen

Aus den Interessen der Benutzer und Wasserzeichenentwickler aus Abschnitt 4.2 werden hier die daraus folgenden Bedingungen an das gesamte Modell und die zu beachtenden Kriterien herausgearbeitet.

Damit möglichst viele Benutzer überall digitale Wasserzeichen verwenden können, bietet es sich an, das Internet als Plattform zu verwenden und von dort die Wasserzeichen bereitzustellen. Damit Anbieter und Kunden zusammenkommen können, wird ein **Client-Server-System für das Internet mit webbasierter Kommunikation** entwickelt. Auf der Serverseite sind die Wasserzeichenentwickler als Anbieter ihrer Wasserzeichen. Die Kunden sind die Benutzer auf der Clientseite.

Die Wasserzeichenentwickler möchten ihre Wasserzeichen gewinnbringend anbieten, um u. a. die Entwicklungskosten finanzieren zu können. Hierzu ist es notwendig, dass die **Kosten zur Bereitstellung eines Client-Server-Systems so gering wie möglich** gehalten werden. Dies ist am ehesten durch möglichst niedrige Serverkosten erreichbar, wofür es notwendig ist, dass der **Server so wenige Aufgaben wie möglich** hat. Somit wird die Rechenlast des Servers gering gehalten, wodurch die Kosten niedrig bleiben. Dies darf jedoch nicht die Nutzung des Systems beeinträchtigen, was bedeutet, dass es mehreren Kunden gleichzeitig möglich sein muss, darauf zuzugreifen.

Neben der Optimierung der Bereitstellungskosten darf die Sicherheit nicht außer Acht gelassen werden. Zur Sicherheit gehört sowohl die Sicherheit der Wasserzeichen, als auch die Sicherheit des gesamten Systems. Wasserzeichenentwickler müssen ausschließen können, dass durch das Anbieten in einem Client-Server-System effizientere Angriffe auf ihre Wasserzeichen ermöglicht werden. Soll die Sicherheit der Wasserzeichen gewährleistet sein, so darf der **Markierungsalgorithmus nicht vom Server herausgegeben** werden, denn wird er auf dem Client ausgeführt oder befindet er sich dort, so ist er öffentlich. Das würde nach Abschnitt 3.4.5 der Stufe *der Angreifer kennt den Markierungsalgorithmus* entsprechen und effizientere Angriffe auf Wasserzeichen ermöglichen.

Soll der Markierungsalgorithmus also nicht auf dem Client ausgeführt werden, so könnte das Medium auf den Server transportiert und dort markiert werden. Der Transfer eines gesamten Mediums gefährdet jedoch die Sicherheit der Originalmedien und bewirkt Einbußen in der Performanz. Diese soll ausreichend schnell bei der Ausführung sein und nicht unnötige Bandbreite sowie unnötigen Speicherplatz verschwenden. Eine gute Performanz kann beispielsweise erreicht werden, wenn die Medien nicht komplett vom Client zum Server übermittelt werden, was auch nicht nötig ist. Für das Markieren mit Wasserzeichen ist es häufig ausreichend, nur die dafür relevanten Medienteile, also **so wenig wie möglich**, **vom Client zum Server zu transportier**en. Dazu muss das Medium zuvor auf dem Client analysiert werden. Dann müssen die entscheidenden Medienteile herausgesucht werden. Für diese Aufgabe benötigt der Client eine **aktive Komponente**.

Das hier entworfene Modell soll später mit der Programmiersprache Java umgesetzt werden, denn diese gilt als sichere Sprache [AdKr03]. In Java gibt es als aktive Komponenten beispielsweise **Applikationen oder Applets**, die hier einen Client darstellen. Die Gegenstücke auf dem Server könnten beispielsweise **Servlets oder Web Services** sein.

Die Clientkomponente muss mit der Serverkomponente kommunizieren um zusammen arbeiten zu können. Hierbei muss die Sicherheit der Kommunikation über das Internet beachtet werden, wodurch der **Einsatz von sicherem HTTP (HTTPS)** bedingt ist.

Die Sicherheit des gesamten Systems ist durch eine sichere Kommunikation noch nicht abgedeckt. Zu diesem Zweck muss darüber hinaus eine **Authentifizierung des Servers** gegenüber dem Client vorgenommen werden. Durch den Einsatz eines signierten Applets kann der Benutzer dem angeforderten Appletcode des Servers vertrauen.

Die aufgestellten Bedingungen sollen später als Grundlage zu einer Bewertung des umgesetzten Client-Server-Modells dienen. Hierfür werden sie in Tabelle 1 zusammengefasst aufgelistet, wobei immer das gewünschte Ziel mit der daraus folgenden Bedingung angegeben ist.

Gewünschtes Ziel	Daraus folgende Bedingung
Von jedem und überall zu verwenden	Client-Server-System im Internet (webbasierte Kommunikation)
Wasserzeichenentwickler möchten möglichst hohen Gewinn erzielen	Bereitstellungskosten eines Client-Server-Systems so gering wie möglich halten
Gewinn nicht unnötig verringern	Server soll möglichst wenige Aufgaben haben und Client muss Aufgaben übernehmen
Effizientere Angriffe auf Wasserzeichen vermeiden	Server darf Wasserzeichenalgorithmus nicht herausgeben
Client soll Aufgaben übernehmen	Client benötigt für seine Aufgaben eine aktive Komponente
Umsetzung mit der Programmiersprache Java	Client als Applikation oder Applet und Server als Servlet oder Web Service realisieren
Bestmögliche Performanz	So wenig wie möglich von Medien an den Server übertragen
Sicherheit der Kommunikation	Einsatz von HTTPS
Sicherheit des Systems	Authentifizierung des Servers

Tabelle 1: Bedingungen für das Client-Server-Modell

5.3 Aufgabenverteilung zwischen Client und Server

Die nächste Frage, die es bei der Umsetzung des Projektes zu behandeln gilt, besteht in der Aufteilung der Wasserzeichenkomponenten in einem Client-Server-System unter Beachtung der herausgearbeiteten Bedingungen: Wie können die Aufgaben eines Wasserzeichenverfahrens, das bereits in Komponenten geteilt vorliegt, an Client und Server verteilt werden? Diese Frage wird nun zunächst allgemein und im nächsten Abschnitt schließlich speziell an einem ausgewählten Beispiel behandelt.

Kapitel 4.3 hat eine Unterteilung von Wasserzeichenverfahren in Komponenten vorgenommen. Dabei wurden für den Markierungsprozess die wesentlichen Schritte *Analyse des Trägermediums und Heraussuchen der zum Markieren relevanten Medienteile*, *Markierungsalgorithmus* und *Zusammenfügen markierter Medienteile mit Originalmedium* angenommen. Möchte man diese Komponenten in einem Client-Server-System ausführen, so müssen sie auf Client und Server verteilt werden. Dabei muss beachtet werden, dass auf dem Client nur sicherheitsirrelevante Funktionalitäten ausgeführt werden sollten, denn die dort ausgeführten Aufgaben sind öffentlich. Aus den in Kapitel 5.2 aufgestellten Bedingungen ergibt sich, dass der Schritt *Analyse des Trägermediums und Heraussuchen der zum Markieren relevanten Medienteile* auf dem Client stattfinden soll. Diese Aufgaben sind nicht sicherheitsrelevant und können deswegen unbesorgt auf dem Client ausgeführt werden. Die herausgesuchten Medienteile werden dann zum

Server transportiert, um dort markiert werden zu können. So verbleibt der *Markierungsalgorithmus*, der eine sicherheitsrelevante Funktionalität darstellt, sicher auf dem Server. Nach der Markierung werden die markierten Medienteile an den Client zurückgegeben. Dieser muss die markierten Medienteile mit dem Originalmedium verbinden (*Zusammenfügen markierter Medienteile mit Originalmedium*) und erhält als Ergebnis ein markiertes Medium. Der beschriebene Ablauf ist in Abbildung 4 dargestellt:

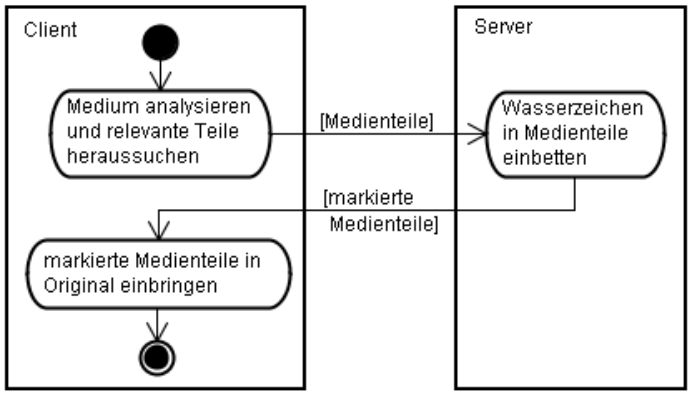

Abbildung 4: Aufgabenverteilung beim Markierungsprozess

Das beschriebene Vorgehen kann auch auf den Ausleseprozess übertragen werden. Dort werden nach Abschnitt 4.3 die Schritte *Analyse des Trägermediums und Heraussuchen der Medienteile, in denen das Wasserzeichen potentiell enthalten sein kann* und *Auslesealgorithmus* unterschieden. Der Client beginnt wie beim Markierungsprozess mit der *Analyse des Trägermediums und dem Heraussuchen der Medienteile, in denen das Wasserzeichen potentiell enthalten sein kann*. Diese Medienteile werden dann an den Server gesendet, der schließlich daraus das Wasserzeichen ausliest (*Auslesealgorithmus*). Die erhaltene Wasserzeichennachricht übergibt der Server an den Client, damit sie dort dem Benutzer angezeigt werden kann. Die beschriebene Aufgabenteilung zwischen Client und Server beim Ausleseprozess ist in Abbildung 5 dargestellt:

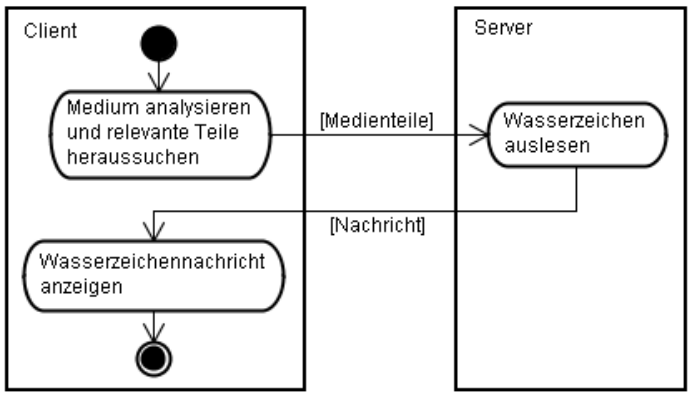

Abbildung 5: Aufgabenverteilung beim Ausleseprozess

Die hier herausgearbeitete Aufgabenteilung zwischen Client und Server ist allgemein für Wasserzeichenverfahren beschrieben. Für ein konkretes Wasserzeichenverfahren muss die Aufgabenteilung zwischen Client und Server individuell untersucht werden, denn durch unterschiedliche Implementierungen der Verfahren kommt es zu verschiedenen Aufgaben die entsprechend zugewiesen werden müssen. In dieser Arbeit wird das Skalenfaktoren-Audiowasserzeichen beispielhaft umgesetzt, wofür im nächsten Abschnitt eine Aufgabenteilung zwischen Client und Server vorgenommen wird.

5.4 Aufgabenverteilung beim Skalenfaktoren-Audiowasserzeichen

Um das Skalenfaktoren-Audiowasserzeichen aus Abschnitt 4.4.1 in einem Client-Server-System umsetzen zu können, müssen zunächst die konkreten Aufgaben von Client und Server abgegrenzt werden. Dazu werden die allgemeinen Ergebnisse aus dem vorigen Abschnitt für dieses Wasserzeichenverfahren konkretisiert. Der Client soll eine MP2-Datei analysieren und die Skalenfaktoren heraussuchen, es wird also *makelist.exe* auf dem Client ausgeführt. Im Anschluss daran sollen die erhaltenen Skalenfaktoren an den Server übergeben werden, denn dieser ist für das Markieren zuständig. Die Aufgaben des Servers sind die Ausführung von *wminstxt.exe* zum Markieren der Skalenfaktoren und das Senden der markierten Skalenfaktoren zurück an den Client. Schließlich wird der Client die markierten Skalenfaktoren durch *link.exe* mit der originalen MP2-Datei verbinden. Diese grobe Beschreibung der Aufgaben inklusive deren Verteilung zwischen Client und Server beim Markierungsprozess werden in Abbildung 6 dargestellt.

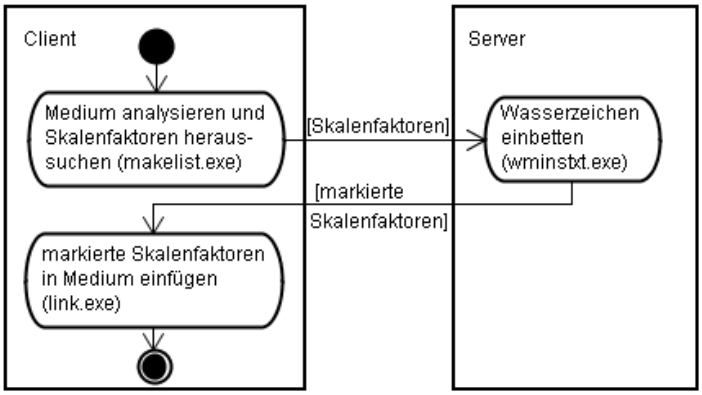

Abbildung 6: Aufgabenverteilung bei einem konkreten Markierungsprozess

Der Ausleseprozess ist vergleichbar mit dem Markierungsprozess. Als erstes werden ebenfalls durch die Ausführung von *makelist.exe* auf dem Client die MP2-Datei analysiert und die Skalenfaktoren herausgesucht, welche dann an den Server gesendet werden. Dieser versucht durch die Ausführung von *wmdet.exe* die Wasserzeichennachricht auszulesen. Anschließend wird die ausgelesene Nachricht an den Client gesendet, damit sie dort dem Benutzer angezeigt werden kann. Abbildung 7 stellt den Ausleseprozess des Skalenfaktoren-Audiowasserzeichenverfahrens in einem Client-Server-Modell dar.

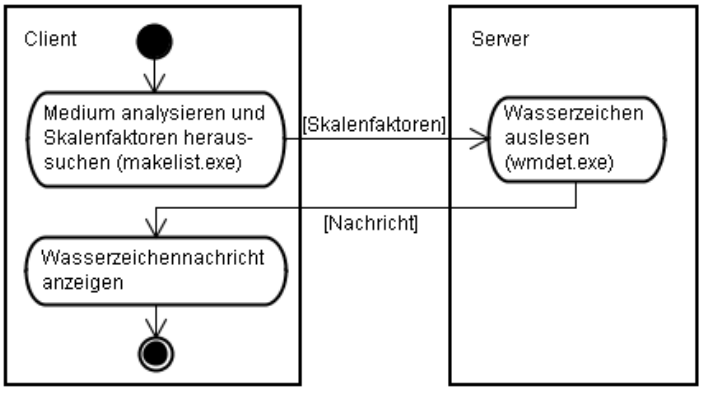

Abbildung 7: Aufgabenverteilung bei einem konkreten Ausleseprozess

Die Aufgabenverteilungen zwischen Client und Server sind nun erarbeitet. Zusammen mit den Bedingungen aus Abschnitt 5.2 werden nachfolgend mögliche Umsetzungen der Komponenten Client und Server beschrieben. Diese Möglichkeiten beziehen sich allgemein auf das entwickelte Client-Server-Modell und beschränken sich nicht nur auf die konkrete Umsetzung des Skalenfaktoren-Audiowasserzeichens.

5.5 Clientkomponente

Die Bedingungen in Abschnitt 5.2 fordern auf dem Client eine aktive Komponente. Aus Abschnitt 5.3 gehen für den Client zum einen beim Markierungsprozess die Aufgaben *Analyse des Trägermediums und Heraussuchen der zum Markieren relevanten Medienteile* und *Zusammenfügen markierter Medienteile mit Originalmedium* und zum anderen beim Ausleseprozess die Aufgabe *Analyse des Trägermediums und Heraussuchen der Medienteile, in denen das Wasserzeichen potentiell enthalten sein kann* hervor. Um diese Aufgaben ausführen zu können, benötigt die Clientkomponente Rechte zum Schreiben und Ausführen auf dem Clientrechner. Dies kann in Java durch aktive Komponenten, beispielsweise als Applikation oder Applet, umgesetzt werden.

5.5.1 Java-Applikation

Java-Applikationen sind Computer-Programme mit vollem Funktionsumfang, die meist auf einer virtuellen Maschine (Java Virtual Machine (JVM)) ausgeführt werden. Sie können daher als lokale Programme auf dem Rechner des Benutzers (Clientrechner) laufen. Für Applikationen gibt es umfangreiche Klassenbibliotheken, so genannte Application Programming Interfaces (API), in denen der Funktionsumfang und auch Sicherheitsmodelle beschrieben sind. Sie unterliegen jedoch erst einmal keinen Sicherheitsrestriktionen, sondern besitzen auf dem Rechner, auf dem sie ausgeführt werden, alle Rechte.

5.5.2 Java-Applet

Ein Benutzer kann im World Wide Web mit Hilfe eines Web-Browsers auf Multimediadokumente zugreifen. So kann auch ein Java-Applet ([Sun] java.applet.Applet) im Browser eines Benutzers aufgerufen und angezeigt werden. Ein Java-Applet ist eine spezielle Java-Applikation, der ebenfalls umfangreiche APIs zur Verfügung stehen und die zum Ausführen neben der virtuellen Maschine einen Browser oder einen anderen Applet-Viewer benötigt. Ein Applet gehört zu den aktiven Dokumenten des WWW, stellt also ein Programm dar, das heruntergeladen und lokal ausgeführt werden kann und das selbst weiß, wie es angezeigt wird. Fordert ein Benutzer ein Applet an, so gibt der Server eine Kopie dieses Programms zurück, mit welcher der Benutzer lokal interagieren kann. Da Applets im Gegensatz zu normalen Applikationen bereits ein Sicherheitskonzept in Form einer so genannten Sandbox, das in Abschnitt 5.8 deutlich wird, vorgeben, sieht der konkrete Modellentwurf des Skalenfaktoren-Audiowasserzeichens auf der Clientseite ein Applet vor. ([Bengel04] Kapitel 3.4.3.1)

5.6 Serverkomponente

Die Aufgaben des Servers bestehen nach Abschnitt 5.3 in der Ausführung des *Markierungsalgorithmus* sowie des *Auslesealgorithmus*. Die Serverkomponente soll ebenfalls mit der Programmiersprache Java umgesetzt werden. Hier bietet sich eine Realisierung als Java-Servlet oder Web-Service an.

5.6.1 Java-Servlet

Ein Java-Servlet ([Sun] javax.servlet.http.HttpServlet) kann als Gegenstück des Applets auf der Serverseite gesehen werden. Das hier entworfene Modell sieht ein Servlet auf der Serverseite vor, da dieses gut mit einem Applet koordiniert werden kann und für die Anforderungen der Arbeit geeignet erscheint. Ein Servlet ist ebenfalls eine spezielle Java Applikation, die zu den dynamischen Dokumenten des WWW gehört und ein Programm darstellt, das sich auf einem Server befindet und bei einer Anfrage eines Clients vom Server aufgerufen und ausgeführt wird. Die Ausgabe wird dynamisch vom Server zusammengestellt als Antwort an den Client zurück gesendet. ([Bengel04] Kapitel 3.4.2.3)

5.6.2 Web Service

Mit Web Services [W3C06] können verteilte Anwendungen realisiert werden. Sie definieren einen Standard des W3C [W3C06], der festlegt, wie unterschiedliche Softwarekomponenten über ein Netzwerk zusammenarbeiten können. Dabei wurde besonderes Augenmerk auf die Plattformunabhängigkeit gelegt – Web Services sollen möglichst hersteller- und systemunabhängig miteinander arbeiten können. Als Verbindungsnetzwerk wurde von Anfang an das Internet mit dem Ziel eingeplant, Anwendungen mit Web Services über dieses Medium zusammenzuschalten. Wichtige Bausteine, aus denen dieser Standard u. a. zusammengesetzt ist, für dessen Funktion, Architektur und Distribution stellen die folgenden W3C-Standards dar: Die Beschreibungssprache Extensible Markup Language (XML), das Internetprotokoll Simple Object Access Protocol (SOAP), das zentrale Register Universal Description, Discovery and Integration (UDDI) und die Service-Beschreibungssprache Web Services Description Language (WSDL). [W3C06]

5.7 Zusammenspiel der Komponenten

Das hier zu entwerfende Modell besteht aus zwei Teilen, einem Client und einem Server. Client- und Serverkomponente müssen bei konkreten Wasserzeichenverfahren jeweils individuell entworfen werden. In dieser Arbeit soll die Clientkomponente in Form eines Applets und die Serverkomponente als Servlet entworfen werden. Auf der Clientseite befindet sich außerdem der Benutzer, der mit dem Applet interagieren können muss. Die Kommunikation und der Transport zwischen dem Applet bzw. dem Client und dem Server, auf dem sich neben dem Servlet weitere notwendige Teile für das Applet befinden, müssen möglich sein. Nachfolgend werden die Interaktion zwischen Benutzer und Applet, sowie die Kommunikation und der Transport zwischen Client und Server beschrieben.

5.7.1 Interaktion zwischen Benutzer und Applet

Damit ein Benutzer mit einem Applet interagieren kann, braucht dieses eine grafische Oberfläche. Java stellt Schnittstellen zum Erstellen grafischer Oberflächen bereit, von denen hier Swing ([Sun] javax.swing) verwendet werden soll. Die Oberfläche soll im Hinblick auf eine Wiederverwendung für mehrere Wasserzeichenimplementationen eine separate Klasse darstellen.

Eine Voraussetzung beim Markieren mit digitalen Wasserzeichen ist immer ein Medium, das markiert werden soll. Der Benutzer muss also in der Appletoberfläche ein Textfeld vorfinden, in das der Pfad zu dem Medium eingetragen werden kann. Des Weiteren muss eine Wasserzeichennachricht eingegeben werden können, die in das Medium eingebettet werden soll und es muss einen Button geben, durch den der Markierungsprozess ausgelöst werden kann.

Zum Auslesen der Wasserzeichennachricht muss der Benutzer den Pfad zu dem markierten Medium angeben können, aus dem die Wasserzeichennachricht ausgelesen werden soll. Dafür muss auf der Appletoberfläche ein Textfeld zur Verfügung stehen. Des Weiteren muss ein Button existieren, durch den der Ausleseprozess gestartet werden kann.

Bei beiden Vorgängen, Markieren und Auslesen, kann es zu Eingabefehlern kommen. Das Modell muss gegenüber diesen eine gewisse Robustheit aufweisen, es sollte also darauf mit einem definierten Zustand reagieren und immer dann, wenn ein Benutzer etwas in der Oberfläche des Applets einträgt, eine Prüfung der Korrektheit der Parameterinhalte durchführen.

Die hier entworfene Interaktion zwischen Benutzer und Applet ist in Abschnitt 6.1.2 implementiert. Drückt der Benutzer einen Button, so soll bei richtig eingegebenen Parametern der gewählte Prozess beginnen. Hierbei ist eine Kommunikation zwischen Client und Server notwendig, die im folgenden Abschnitt beschrieben wird.

5.7.2 Kommunikation und Dateitransport zwischen Applet und Servlet

Die Kommunikation der Komponenten Client und Server soll webbasiert sein und wird zunächst vom Client veranlasst. Der Client sendet dafür eine Anforderung (request) an den Server, in der zuvor notwendige Parameter gesetzt werden müssen, wie beispielsweise einen Parameter *action*, durch den angegeben werden kann, welche Aktion vom Servlet durch das Applet angesprochen wird. In Abschnitt 6.1.3 ist die Implementierung mit einer detaillierten Beschreibung zu finden.

Neben der einfachen Kommunikation, also den direkten Anfragen des Applets an das Servlet, muss auch der Transport von Dateien zwischen Client und Server ermöglicht werden. Dieser Transport erfolgt über eine HTTPS-Verbindung, für deren Realisierung Client und Server SSL-Verschlüsselung unterstützen müssen. In Abschnitt 6.1.4 ist der Dateitransport zwischen Client und Server implementiert und dessen Voraussetzungen beschrieben.

5.8 Die Ziele Sicherheit und Performanz

Die Ziele Sicherheit und Performanz stellen den Hauptzweck der vorliegenden Arbeit dar. In der Umsetzung dieses Projekts wird die Programmiersprache Java verwendet, denn diese gilt, wie in Abschnitt 5.2 angesprochen, als sichere Programmiersprache. Hier interessiert besonders die Sicherheit beim Einsatz von Java-Applets, die im nächsten Abschnitt beschrieben wird. Um benötigte Rechte zu bekommen müssen Applets signiert werden. Signierte Applets sind Thema des Abschnittes 5.8.2. In den Abschnitten 5.8.2 und 5.8.3 werden die Sicherheit des gesamten Modells und der Wasserzeichenverfahren untersucht. Die Performanz des gesamten Ablaufs ist Thema in Abschnitt 5.8.5.

5.8.1 Java-Applets und Sicherheit

Die Grundlage für einen sicheren Einsatz von Java-Applets, bietet das Sprachmodell von Java. Java gilt als sichere Sprache, da sie selbstständig viele Kontrollen einsetzt und dadurch dem Programmierer viel Arbeit erspart. Java beinhaltet darüber hinaus einige sicherheitsrelevante Merkmale, zu denen unter anderem die virtuelle Maschine, eine strenge Typisierung, Exceptions, die Garbage Collection, strenge Arraygrenzen sowie die Verwendung von Referenzen statt Pointern zählen. Diese Merkmale machen die Sprache robust, tolerant gegenüber Programmierfehlern und damit unanfälliger für Seiteneffekte. [AdKr03]

Applets sind aktive Komponenten, die ein großes Risiko darstellen würden, wenn sie von vornherein auf dem Clientrechner alle Rechte besitzen würden, also auch auf kritische Systembereiche zugreifen könnten, da hierdurch für den Rechner wichtige Dateien manipuliert oder gar gelöscht werden und zudem vertrauenswürdige Daten ausgelesen werden könnten. Aus diesem Grund wird erst einmal jeder geladene Applet-Codeteil vom Browser als nicht vertrauenswürdig eingestuft. Das Sicherheitskonzept von Java setzt einen abgeschlossenen gesicherten Bereich, eine so genannte Sandbox, um, in der nicht vertrauenswürdiger Code ausgeführt werden kann [Pilz97]. Dabei gibt es keine Lücke nach außen. Es wird unter anderem jeder Zugriff auf Systemressourcen verhindert, es ist kein Zugriff auf den Hauptspeicher außerhalb des der Sandbox zugewiesenen Adressbereiches möglich und es können keine Operationen auf Dateiebene durchgeführt werden. Weitere unerlaubte Aktionen, deren Ausführung eine `SecurityException` auslösen, sind in [Kopp98] nachzulesen.

Wie bereits in Abschnitt 5.5 angesprochen sind die genannten Restriktionen durch das Konzept der Sandbox gegeben und gelten speziell für Applets, die aus diesem Grund in der vorliegend beschriebenen Umsetzung des Projektes als aktive Clientkomponente eingesetzt werden.

Das Sandbox-Modell besteht hauptsächlich aus den Komponenten *Class Loader Architektur*, *Class File Verifier* und *Security Manager*, durch die unter anderem fehlerhafter Code vermieden wird. Details der Komponenten können in [Fitzinger03] nachgelesen werden. Das Sandbox-Modell ist vollständig von [Sun] definiert, muss jedoch von den Entwicklern bzw. Browser-Herstellern implementiert werden. Dieses ist zwar sicher, jedoch auch sehr restriktiv. Um die Einschränkungen durch die Verwendung einer Sandbox für vertrauenswürdige Applets aufzuheben, können signierte Applets eingesetzt werden [Fitzinger03]. Der Einsatz und die Möglichkeiten signierter Applets werden im nächsten Abschnitt erklärt.

5.8.2 Signierte Applets

Eine digitale Signatur stellt eine digitale Unterschrift dar. Sie stellt sowohl sicher, dass das Applet von einer bestimmten Quelle stammt (Authentizität und Verbindlichkeit), als auch, dass das Applet auf dem Weg von der Quelle, dem Server, zum Benutzer nicht verändert worden ist (Integrität). Ein Browser fragt beim ersten Aufruf eines signierten Applets beim Benutzer nach, ob er dieses ausführen darf. Bei weiteren Aufrufen des gleichen Applets liegt es an der Einstellung des Browsers, ob dieser jedes Mal nachfragt, wenn er ein signiertes Applet aufruft, oder ob der Benutzer auch angeben kann, dass einem signierten Applet bei weiteren Aufrufen immer vertraut wird, ohne jedes Mal erneut nachzufragen. Ein signiertes Applet, dessen Quelle vom Benutzer als vertrauenswürdig akzeptiert wird, wird als vertrauenswürdig eingestuft und erhält damit erweiterte Rechte [Fitzinger03]. Diese erweiterten Rechte sind vergleichbar mit denen, die eine Applikation von vorneherein besitzt, ohne dass sie vorher vom Benutzer akzeptiert werden muss. Nur durch eine Signatur besitzt ein Applet notwendige Rechte zum Schreiben, Erstellen und Ausführen von Dateien.

Bei der Umsetzung dieses Projekts muss das Applet Dateien vom Server herunterladen und auf dem Client speichern und außerdem Executables ausführen. Es ist folglich notwendig, die verwendeten Applets zu signieren und natürlich muss der Benutzer sie auch als vertrauenswürdig akzeptieren. Der Einsatz signierter Applets erfolgt in Abschnitt 6.1.1.

5.8.3 Sicherheit des Modells

Die Sicherheit eines Systems ergibt sich, wie in Abschnitt 2.3.3 geschrieben, aus dem Zusammenspiel der Sicherheit jeder Komponente des Systems. Die Sicherheit eines Client-Server-Systems hängt also von der Sicherheit der Komponenten Client, Server und Netzwerk ab. Client und Server müssen sich jeweils nach außen in Richtung des Netzes schützen, beispielsweise mit einer Firewall. Die Sicherheit von Client und Server wird hier als gegeben vorausgesetzt. Die Sicherheit des Netzes wird in diesem Modell durch eine gesicherte HTTPS-Verbindung umgesetzt.

HTTPS ist bereits in Abschnitt 3.4.4 beschrieben. Durch den Einsatz von HTTP über SSL sind die Authentifikation der Kommunikationspartner Client und Server, eine vertrauliche Datenübertragung und die Integrität der transportierten Nachrichten und Daten gewährleistet. HTTPS ermöglicht den Aufbau einer sicheren SSL-Verbindung, bei dem eine verschlüsselte Kommunikation stattfindet. In diesem Modell reicht die einseitige Authentifikation des Servers dem Client gegenüber. Dafür muss der Server ein Zertifikat besitzen, das sichere Verbindungen akzeptiert, und entsprechend konfiguriert sein.

Durch den Einsatz signierter Applets (siehe vorheriger Abschnitt) ist, zusätzlich zur sicheren Übertragung, die Authentizität, Verbindlichkeit und Integrität des Applet-Codes selbst nachgewiesen. Der Applet-Code wird vom Server heruntergeladen und muss vom Benutzer akzeptiert werden, damit das Applet notwendige Rechte bekommt.

Der Einsatz von HTTPS gewährleistet eine sichere Verbindung zwischen Client und Server. Die Signatur der Applets lässt den Benutzer zusätzlich entscheiden, ob er dem Server insoweit vertraut, dass er dem Applet die nötigen Rechte zum Schreiben und Ausführen gibt. So wird neben der sicheren Verbindung die Authentizität und Verbindlichkeit des Applet-Codes selbst nachgewiesen, denn es könnte ja sein, dass der Server durch unbefugte Eingriffe modifiziert worden ist und das eigentlich angeforderte Applet

durch ein beschädigtes ausgetauscht worden ist. Durch die Signatur des Applets wird dem Benutzer eine zusätzliche Sicherheit geboten. Neben der zusätzlichen Sicherheit durch die Signatur des Applets, ist diese auch aus dem Grund der Rechte notwendig, denn nur ein akzeptiertes signiertes Java-Applet hat Rechte zum Schreiben und Modifizieren von Dateien und Medien.

5.8.4 Sicherheit der Wasserzeichen

Die Entwicklung digitaler Wasserzeichen ist noch nicht so weit, dass sie dem Kerckhoffs-Prinzip entspricht [CaFoFu05]. Das bedeutet, dass ein potentieller Angreifer, der an den Markierungsalgorithmus gelangt, nach der in Abschnitt 3.4.5 definierten Stufe *der Angreifer kennt den Markierungsalgorithmus* effizientere Angriffe auf Wasserzeichen durchführen kann. Die daraus folgende Konsequenz für das hier umgesetzte Modell ist, dass der Markierungsalgorithmus sicher auf dem Server bleibt, da alle an den Client gesendeten Informationen als öffentlich angesehen werden müssen.

Hierfür ist es notwendig, dass der Client weiß, welche Teile für den Wasserzeichenalgorithmus gebraucht werden, denn diese müssen von ihm herausgesucht werden. Dadurch weiß der Client, in welche Medienteile das Wasserzeichen eingebettet wird. Um dieser Schwachstelle, dass der Client etwas über das Wasserzeichenverfahren kennt, entgegen zu wirken, könnte die Aufgabe des Clients insofern abgeändert werden, dass mehr Medienteile, als eigentlich zum Markieren notwendig wären, herausgesucht und an den Server geschickt werden. Dadurch wird es für den Client, aber auch für mögliche Angreifer unmöglich zu wissen, welche Medienteile das digitale Wasserzeichen enthalten. Jedoch wird diese Überlegung unnötig, denn der Client ist nach der Markierung im Besitz des Originalmediums und einer markierten Kopie. Durch deren Vergleich kann ebenfalls festgestellt werden, in welchen Medienteilen das Wasserzeichen eingebettet ist. Diese Kenntnis würde in der Abstufung der Effizienz der Angriffe auf Wasserzeichen aus Abschnitt 3.4.5 der zweiten Ebene entsprechen – *der Angreifer hat ein markiertes Medium*. Eine denkbare Gegenmaßnahme für dieses Problem sind Maskierungsverfahren, die als Teil des Wasserzeichenverfahrens angewendet werden können. Sie fügen dem Medium zusätzlich zum eigentlichen Wasserzeichen ein Pseudo-Rauschen hinzu, damit bei einem direkten Vergleich von originalem und markiertem Medium nicht auf die Veränderungen durch das Wasserzeichen selbst geschlossen werden kann. Denn, wie in Abschnitt 3.4.5 am Beispiel des LSB-Wasserzeichens beschrieben wurde, können durch das Wissen über die für den Wasserzeichenalgorithmus relevanten Teile effizientere Angriffe auf das Wasserzeichenverfahren ermöglicht werden. Beim Einsatz von Maskierungsverfahren muss jedoch immer bedacht werden, dass zusätzlich zu den Veränderungen durch das Wasserzeichen noch weitere Störungen hinzugefügt werden und dadurch die Qualität des Mediums beeinträchtigt werden kann.

5.8.5 Performanz des gesamten Ablaufs

Die Performanz des gesamten Ablaufs muss im Vergleich der Dauer des Markierens bei Verwendung des hier umgesetzten Client-Server-Modells zu der Dauer des Markierens bei lokaler Ausführung betrachtet werden.

Der wesentliche Schritt, der die Dauer bei Verwendung des Client-Server-Modells verlängert, ist der Datentransfer zwischen Client und Server. Teile des Wasserzeichenverfahrens müssen durch den Client heruntergeladen werden. An dieser Größe kann nichts verändert werden. Die Medienteile, die vom Client zum Server transportiert wer-

den müssen, sind dagegen abhängig von der Größe des gesamten Mediums. Außerdem hängt es vom Wasserzeichenverfahren und dessen Implementierung ab, wie viel Prozent des eigentlichen Mediums an den Server übertragen werden müssen. Diese Eigenschaft des Wasserzeichenverfahrens wirkt sich auf dessen Eignung für die Umsetzung in dem hier entwickelten System aus.

Des Weiteren spielt die Rechenlast des Servers eine entscheidende Rolle, die für einen Wasserzeichenprozess möglichst gering gehalten werden soll. Das bedeutet, dass der Client soviel Aufgaben wie möglich übernehmen soll, zu denen, neben der Analyse des Mediums und dem Heraussuchen der zum markieren notwendigen Medienteile, ebenfalls Aufgaben wie beispielsweise notwendige Formatwandlungen zählen, die die Rechenlast des Servers nur unnötig beanspruchen würden.

Beeinträchtigungen der Performanz können außerdem durch die Verwendung von SSL auftreten. Diese sind jedoch so gering, dass sie vernachlässigt werden können. Auftretende Performanzeinbußen durch das Packen von Dateien vor dem Versenden und das notwendige Entpacken der gesendeten Dateien können dagegen nicht unangesprochen bleiben und werden in der abschließenden Diskussion in Abschnitt 6.3 genauer betrachtet.

Kapitel 6

Implementierung eines Client-Server-Modells zum Markieren von Medien

In diesem Kapitel wird das zuvor entworfene Client-Server-Modell, mit welchem digitale Medien im Internet sicher mit digitalen Wasserzeichen markiert werden können, implementiert. Dabei sollen die im Entwurf erarbeiteten Bedingungen (siehe Abschnitt 5.2) eingehalten werden. Auf der Clientseite soll ein Applet und auf der Serverseite ein Servlet implementiert werden. Die Implementierung dieses Applet-Servlet-Konstrukts wird zunächst allgemein beschrieben, bevor als konkretes Wasserzeichenverfahren das Skalenfaktoren-Audiowasserzeichen umgesetzt wird. Hierfür wird betrachtet, wie ein Benutzer auf Clientseite mit dem Applet interagieren kann. Außerdem sind die Kommunikation und das Zusammenspiel von Client und Server bzw. Applet und Servlet von Interesse. Das vorgestellte Modell soll die Performanz des Transports digitaler Medien über das Internet und die Sicherheit des Wasserzeichenalgorithmus sowie die Sicherheit der Originalmedien berücksichtigen. Die Umsetzung dieser in Abschnitt 5.8 erarbeiteten Ziele wird in vorliegendem Kapitel integriert. Da für jedes Wasserzeichenverfahren andere Anforderungen und Umsetzungsmöglichkeiten gelten, wird abschließend beispielhaft ein konkretes Wasserzeichenverfahren, das Skalenfaktoren-Audiowasserzeichen, umgesetzt.

6.1 Zusammenspiel der Komponenten

Das hier implementierte Modell besteht aus zwei Teilen, einem Client und einem Server. Dabei wird die Clientkomponente in Form eines signierten Applets umgesetzt. Auf der Clientseite befindet sich der Benutzer, der mit dem Applet interagieren können muss. Die Kommunikation und der Dateitransport zwischen dem Applet bzw. dem Client und dem Server, auf dem sich neben dem Servlet weitere notwendige Teile für das Applet befinden, müssen möglich sein. Nachfolgend werden signierte Applets, die Interaktion zwischen Benutzer und Applet, sowie die Kommunikation und der Dateitransport zwischen Client und Server beschrieben.

6.1.1 Signierte Applets

Bei der Umsetzung dieses Projekts muss das Applet Dateien vom Server herunterladen und auf dem Client speichern und außerdem Executables ausführen, deswegen müssen signierte Applets eingesetzt werden. Dafür müssen zunächst alle zu dem Applet gehörende Dateien in einer Java-Archiv-Datei (JAR-Datei) verpackt und diese Datei dann signiert werden. Zum Signieren von Dateien stellt Java *jarsigner.exe* zur Verfügung. Durch den Aufruf *jarsigner.exe -keystore Henrich_Certificate.pfx -storetype pkcs12 WatermarkingApplet.jar alias -v* wird das Java-Archiv *WatermarkingApplet.jar* mit dem Zertifikat *Henrich_Certificate.pfx* signiert.

JAR-Dateien können durch das Attribut `archive` des `Applet`-Tags wie folgt in einem HTML-Dokument eingesetzt werden:

```
<applet code=„classpath.WatermarkingApplet“ archive="WatermarkingApplet.jar"> </applet>
```

Quelltext 1: HTML-Applet-Tag

Wird im Browser ein HTML-Dokument aufgerufen das Quelltext 1 enthält und ist die zugehörige JAR-Datei (*WatermarkingApplet.jar*) signiert, jedoch beim Browser noch nicht als vertrauenswürdig akzeptiert, so fragt der Browser nach, ob er dem signierten Applet vertrauen soll. Akzeptiert der Benutzer diese Anfrage, so hat dies als Konsequenz, dass das Applet geladen und ausgeführt wird und die nötigen Rechte zum Schreiben und Ausführen von Dateien besitzt.

6.1.2 Interaktion zwischen Benutzer und Applet

Wie bereits in Abschnitt 5.7.1 angesprochen, soll die Oberfläche des Applets mit Swing umgesetzt werden und im Hinblick auf eine Wiederverwendung für mehrere Wasserzeichenimplementationen als separate Klasse, `WatermarkingAppletGUI`, programmiert werden. Die implementierende Klasse, die das konkrete Wasserzeichenverfahren umsetzt, muss lediglich den Informationstext über das Wasserzeichenverfahren selbst setzen, siehe Abbildung 8. Die eingegebenen Parameter, die bereits beschrieben wurden, können von der implementierenden Klasse selbstverständlich verwendet werden.

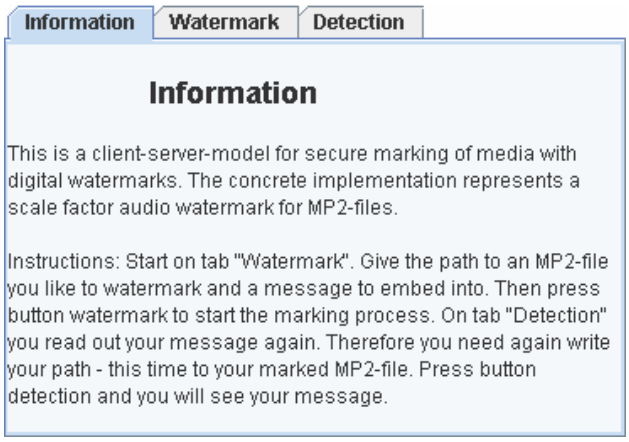

Abbildung 8: Informationsregister

Beim Markieren mit digitalen Wasserzeichen muss vom Benutzer in der Appletoberfläche der Pfad zu dem zu markierenden Medium und eine Wasserzeichennachricht, die in das Medium eingebettet werden soll, eingegeben werden. Abbildung 9 zeigt die Oberfläche eines Appletteils, bei der Pfad und Wasserzeichennachricht eingegeben werden können. Sind diese beiden Textfelder korrekt ausgefüllt, so kann der Markierungsprozess durch den Button *Watermark* ausgelöst werden.

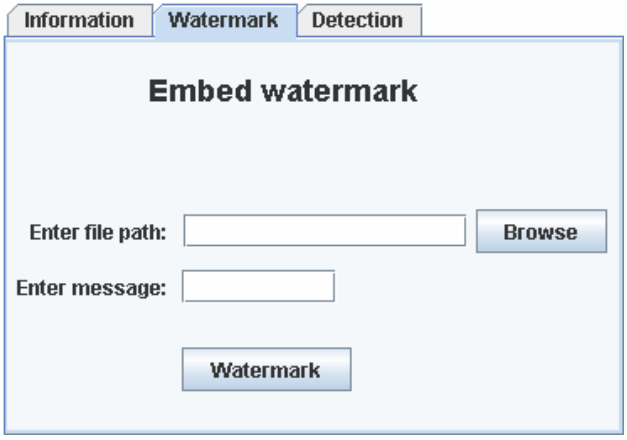

Abbildung 9: Markierungsregister

Zum Auslesen der Wasserzeichennachricht muss der Benutzer den Pfad zu dem markierten Medium angeben. Ist der Pfad korrekt eingegeben, kann durch den Button *Detection* der Ausleseprozess gestartet werden. Abbildung 10 zeigt die Oberfläche des Appletteils zum Auslesen der Wasserzeichennachricht.

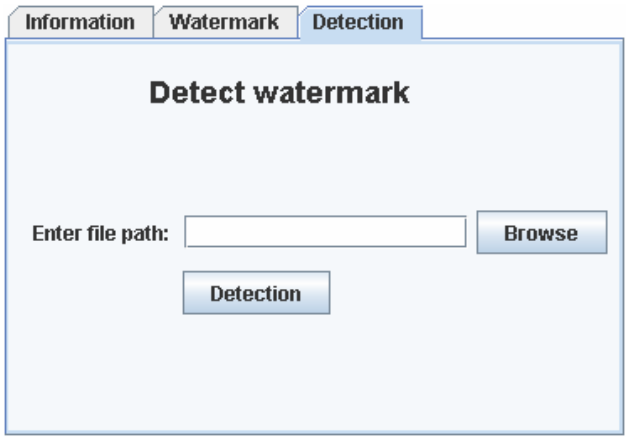

Abbildung 10: Ausleseregister

Wie zu erkennen ist, kann es bei beiden Vorgängen zu Eingabefehlern kommen. Das System muss gegenüber diesen eine gewisse Robustheit aufweisen, es sollte also darauf mit einem definierten Zustand reagieren. In Java steht zum Abfangen von Ereignissen, wozu jegliche Nutzerinteraktion, also Eingaben, Betätigen von Buttons, etc. zählen, das Interface `ActionListener` mit der Methode `actionPerformed(ActionEvent event)` zur Verfügung. Sowohl beim Markieren als auch beim Auslesen muss ein korrekter Pfadname eingetragen werden. Zum Prüfen dieser Eingabe wird in der Methode `actionPerformed` aus dem angegebenen Pfad ein `File` erstellt und dieses auf seine Existenz geprüft. Der Aufruf sieht in Java folgendermaßen aus: `if(new File(embeddingFilePath.getText()).exists())`, wobei `embeddingFilePath`

den eingegebenen Inhalt des Textfeldes *Enter file path* der vorigen beiden Abbildungen darstellt. Durch diese Abfrage kann schnell auf falsche Pfadangaben reagiert und späteren unnötigen Analyseversuchen von nicht existierenden Medien vorgebeugt werden. Die Prüfung, ob beim Markierungsprozess eine Nachricht eingegeben wurde, mittels Überprüfung des Parameters *message* auf eine Mindestlänge.

Die Prüfung der Parameterinhalte auf Korrektheit wird immer dann angewandt, wenn ein Benutzer etwas in der Oberfläche des Applets einträgt. Sie wird im Folgenden nicht jedes Mal angesprochen, sondern als vorhanden angesehen und vorausgesetzt.

Drückt der Benutzer einen Button, so soll bei richtig eingegebenen Parametern der gewählte Prozess beginnen. Hierbei ist eine Kommunikation zwischen Client und Server notwendig, die im folgenden Abschnitt beschrieben wird.

6.1.3 Kommunikation zwischen Client und Server

Die Kommunikation der Komponenten Client und Server ist webbasiert implementiert und wird, wie in Abschnitt 5.7.2 angesprochen, zunächst vom Client veranlasst. Der Client bzw. das Applet sendet dafür eine `HttpServletRequest` an den Server. In dieser Anfrage setzt das Applet zur Bearbeitung notwendige Parameter, wobei für die hier durchgeführte Implementierung immer der Parameter *action* gesetzt ist, der angibt, welche Aktion vom Servlet durch das Applet angesprochen wird.

Alle ankommenden Anfragen werden an eine selbst erstellte Methode weitergeleitet: `doProcess(HttpServletRequest request, HttpServletResponse response)`. Diese unterscheidet die eingehende `HttpServletRequest request` des Applets nach der gewählten Aktion durch Vergleich des Parameters *action* und ruft entsprechende Methoden zur Weiterverarbeitung auf. Der Vergleich, ob der *action* beispielsweise mit *watermark* übereinstimmt, es sich also um dir Aktion Markieren handelt, geschieht folgendermaßen: `WATERMARK.equals(request.getParameter(ACTION))`. Diese Aktion wurde durch das Drücken des Buttons *Watermark* auf dem Applet ausgelöst. Durch eine if-else-Anweisung in der Methode `doProcess(HttpServletRequest request, HttpServletResponse response)` kann nach den verschiedenen Aktionen unterschieden und durch Aufrufe der entsprechenden Methoden die angeforderten Funktionalitäten ausgeführt werden.

6.1.4 Dateitransport zwischen Client und Server

Im Übrigen müssen auch Anfragen behandelt werden, bei denen Dateien vom Client an den Server gesendet werden. Dieser Transport von Medien vom Client an den Server erfolgt über eine HTTPS-Verbindung. Hierfür muss zunächst eine `HttpsURLConnection` aufgebaut werden, die das Applet über den URL des Servers initiieren kann. Der verwendete URL sollte HTTPS unterstützen, also mit *https://* beginnen. Um schließlich Medien über diese Verbindung senden zu können, müssen zuvor einige Parameter gesetzt werden. Quelltext 2 zeigt die Methode `sendByURLConnection(File source, URL destination)`, bei der eine `HttpsURLConnection` vom Client zum Server aufgebaut wird und die nötigen Parameter eingestellt werden, damit schließlich Dateien vom Client an den Server gesendet werden können.

Alle aufgeführten Quelltexte sind lediglich Programmausschnitte zum Veranschaulichen und stellen keinen Anspruch auf Vollständigkeit. Beispielsweise ist kein Exception-Handling berücksichtigt.

```
protected void sendByURLConnection(File source, URL destination) {

  //open HttpsURLConnection to server
  HttpsURLConnection connection = (HttpsURLConnection) destination.openConnection();

  //inform the connection that output will be sent and input accepted
  connection.setDoInput(true);
  connection.setDoOutput(true);
  connection.setAllowUserInteraction(true);

  //don't use a cached version of URLConnection
  connection.setUseCaches(false);
  connection.setDefaultUseCaches(false);

  //specify the content type to send binary data
  connection.setRequestProperty ("Content-Type", "application/octet-stream");
  connection.setRequestMethod("POST");

  //open streams
  connection.connect();
  OutputStream outputStream = connection.getOutputStream();
  BufferedOutputStream outputToServlet = new BufferedOutputStream(outputStream);

  //send file at source to the servlet
  URL urlOnClient = source.toURL();
  BufferedInputStream inputFromApplet=new BufferedInputStream(urlOnClient.openStream());

  //copy inputFromApplet to outputToServlet
  int b;
  while((b = inputFromApplet.read()) != -1) {
    outputToServlet.write(b);
  }

  //close streams
  outputToServlet.flush();
  outputToServlet.close();
  inputFromApplet.close();

}
```

Quelltext 2: sendByURLConnection

Der umgekehrte Weg vom Server zum Client ist über ein eigenständiges Herunterladen der Dateien vom Server realisiert, da dies weniger Aufwand darstellt, als der explizite Aufbau einer `HttpsURLConnection`. Quelltext 3 zeigt die Methode `download(URL source, String destination)`, die eine Möglichkeit zeigt, wie das Applet Dateien vom Server herunterladen kann. Bei dieser Methode kennt der Client den URL des Servers zu den benötigten Dateien und kann deswegen die Dateien eigenständig herunterladen ohne zuvor explizit eine `HttpsURLConnection` aufgebaut zu haben, denn die Verbindung wird von dem Applet selbst erzeugt. Dieses Vorgehen stellt weniger Programmieraufwand dar und wird deswegen beim Herunterladen von Dateien vom Server verwendet. Die Methode `download(URL source, String destination)` ersetzt jedoch nicht die Methode `sendByURLConnection(File source, URL destination)`, bei der explizit eine `HttpsURLConnection` wird, denn beim Hochladen von Dateien vom Client auf den Server ist diese vereinfachte Vorgehensweise nicht möglich da der Server nicht selbstständig Dateien vom Client laden kann, sondern diese gesendet werden müssen. Der in der Methode `download(URL source, String destination)` übergebene Parameter `source` stellt dabei den URL zur vom Client angeforderten Datei dar. Er sollte natürlich ebenfalls HTTPS unterstützen, also mit *https://* beginnen, damit die Sicherheit der Verbindung gewährleistet ist. HTTP würde ebenfalls funktionieren, jedoch ist dann keine sichere Verbindung über SSL gegeben.

```
protected void download(URL source, String destination) {

  //open streams
  BufferedInputStream inputStream = new BufferedInputStream(source.openStream());
  BufferedOutputStream outputStream = new BufferedOutputStream(new
                                                    FileOutputStream(destination));

  //copy inputStream to outputStream
  int b;
  while((b = inputStream.read()) != -1) {
    outputStream.write(b);
  }

  //close streams
  outputStream.flush();
  outputStream.close();
  inputStream.close();

}
```

Quelltext 3: download

6.2 Konkretes Beispiel: Skalenfaktoren-Audiowasserzeichen

Die Komponenten des Skalenfaktoren-Audiowasserzeichens wurden bereits in Abschnitt 4.4.1 beschrieben und in Abschnitt 5.4 wurde eine Aufgabenverteilung zwischen Client und Server vorgenommen. In diesem Abschnitt sollen die gewonnenen Ergebnisse in einen Gesamtablauf eines Client-Server-Systems integriert werden. Dafür werden im nächsten Abschnitt zuerst die zur Implementierung notwendigen Methoden beschrieben. Abschnitt 0 legt den Markierungsprozess dar, beginnend mit dem Heraussuchen der zum Markieren relevanten Medienteile auf dem Client und beschreibt dann das eigentliche Markieren auf dem Server. Außerdem werden die abschließenden Schritte des Clients beschrieben, bei denen das Originalmedium mit den markierten Medienteilen vereinigt wird. Damit ist der Markierungsprozess vollständig. Im Anschluss daran werden in Abschnitt 6.2.3 die detaillierten Schritte des Clients und des Servers zum Auslesen des Wasserzeichens beschrieben.

6.2.1 Zur Implementierung notwendige Methoden

Für die Implementierung des Markierungs- und Ausleseprozesses sind verschiedene Methoden notwendig. Dies sind `executeCommand`, `PackAndUnpack.unpackOneFile` und `PackAndUnpack.pack`, die im Folgenden näher betrachtet werden.

Wie bereits erwähnt, müssen bei der Umsetzung des Skalenfaktoren-Audiowasserzeichens in einem Client-Server-Modell unter anderem die Executables *makelist.exe*, *wminstxt.exe*, *link.exe* sowie *wmdet.exe* ausgeführt werden. Zum Ausführen von Executables in Java dient die Methode `executeCommand(String command)` aus Quelltext 4, die Befehle (`command`) der Kommandozeile, also auch Executables, ausführen kann.

```
protected void executeCommand(String command) {

  //built process
  ProcessBuilder processBuilder = new ProcessBuilder(command);
  processBuilder.directory(new File(pathOnClient));

  //start process
  Process process = processBuilder.start();
  process.waitFor();

}
```

Quelltext 4: executeCommand

Der Dateitransfer zwischen Client und Server findet über ZIP-Dateien statt. Es werden also sowohl die herausgesuchte Skalenfaktoren-Datei vom Client als auch die markierte Skalenfaktoren-Datei und die Teile des Wasserzeichenverfahrens, die der Client vom Server herunterlädt, als ZIP-Dateien gepackt transportiert. Schließlich müssen diese vor einer weiteren Verwendung auch wieder entpackt werden. Für beide Prozesse stellt die Klasse `PackAndUnpack` statische Methoden zum Packen (`pack`) und Entpacken (`unpackOneFile`) bereit.

Mit der Methode `PackAndUnpack.pack(String pathToArchive, String[] filesToPack)` in Quelltext 5 können Dateien im ZIP-Format gepackt werden. Die Parameter der Methode erwarten eine Datei (`filesToPack`), die gepackt werden soll, und den Pfad, unter dem die Datei zu finden ist (`pathToArchive`).

```
public static void pack(String pathToArchive, String[] filesToPack) {

  String packedFile = pathToArchive + filesToPack[0] + ".zip";
  int read = 0;
  FileInputStream inputStream;
  byte[] data = new byte[1024];

  //join zip archive with stream
  ZipOutputStream outputStream = new ZipOutputStream(new FileOutputStream(packedFile));

  //set compression method
  outputStream.setMethod(ZipOutputStream.DEFLATED);

  //add individual entries
  for(int i = 0; i < filesToPack.length; i++) {
    //make an entry for new file
    ZipEntry entry = new ZipEntry(filesToPack[i]);
    inputStream = new FileInputStream(pathToArchive + filesToPack[i]);

    //add new entry to archive
    outputStream.putNextEntry(entry);

    //add data to new entry
    while((read = inputStream.read(data, 0, 1024)) != -1) {
      outputStream.write(data, 0, read);
    }

    //finish new entry
    outputStream.closeEntry();
    inputStream.close();
  }

  outputStream.close();
}
```

Quelltext 5: PackAndUnpack.pack

Die Funktionsweise von der Methode `PackAndUnpack.unpackOneFile(String path, String[] archives, String fileName)` ist in Quelltext 6 zu sehen. Die angegebenen Parameter geben an, welche Datei entpackt werden soll (`filename`), in welchem Archiv sie sich befindet (`archives`) und schließlich den Speicherort dieses Archivs befindet (`path`).

```
public static void unpackOneFile(String path, String[] archives, String fileName) {

  if(archives.length == 1) {
    ZipFile zipFile = new ZipFile(archives[0]);

    //save every entry in zipFile
    for(Enumeration<? extends ZipEntry> e = zipFile.entries(); e.hasMoreElements(); ) {
      ZipEntry targetEntry = e.nextElement();

      if(targetEntry.getName().equals(fileName)) {
        saveEntry(path, zipFile, targetEntry);
      }

    }

  }

}

public static void saveEntry(String pathToFile, ZipFile zipFile, ZipEntry targetEntry )
{

  File file = new File(pathToFile, targetEntry.getName());

  if (targetEntry.isDirectory()) {
    file.mkdirs();
  } else {
    //open streams
    InputStream inputStream = zipFile.getInputStream(targetEntry);
    BufferedInputStream bufferedInputStream = new BufferedInputStream(inputStream);
    new File(file.getParent()).mkdirs();
    FileOutputStream fileOutputStream = new FileOutputStream(file);
    BufferedOutputStream bufferedOutputStream = new
                                          BufferedOutputStream(fileOutputStream);

    //copy bufferedInputStream to bufferedOutputStream
    final int EOF = -1;
    for (int c; (c = bufferedInputStream.read()) != EOF; ) {
      bufferedOutputStream.write( (byte)c );
    }

    //close streams
    bufferedOutputStream.close();
    fileOutputStream.close();
  }

}
```

Quelltext 6: PackAndUnpack.unpackOneFile

6.2.2 Markierungsprozess

Der Markierungsprozess eines Wasserzeichens beginnt mit der *Analyse des Trägermediums und dem Heraussuchen der zum Markieren relevanten Medienteile*. Dabei wird das jeweils verwendete Medium dem Applet vom Benutzer mitgeteilt, siehe Abschnitt 6.1.2. Diese Aufgaben werden auf dem Client ausgeführt und die herausgesuchten Medienteile an den Server gesendet. Der Server markiert die Medienteile (*Markierungsalgorithmus*) und sendet sie zurück an den Client. Der Client verbindet schließlich die markierten Medienteile mit dem Originalmedium (*Zusammenfügen markierter Medienteile mit Originalmedium*). Die einzelnen Schritte des Markierungsprozesses sind im Folgenden detailliert aufgelistet und in Abbildung 11 zusammenfassend dargestellt.

Analyse und Heraussuchen (Applet auf Client)

1. Das Applet wird geöffnet und muss vom Benutzer akzeptiert werden, damit es die nötigen Rechte zum Schreiben und Ausführen hat.
2. Das Applet erwartet vom Benutzer die Eingaben einer Wasserzeichennachricht (beispielsweise *MESSAGE*) und eines Pfades zu einer MP2-Audiodatei (*PfadZurDatei/audiofile.mp2*).
3. Das Applet lädt die Datei *makelist.zip* vom Server herunter (`download()`). In *makelist.zip* befinden sich *makelist.exe* und *bitsplus.txt*.
4. *Makelist.exe* und *bitsplus.txt* werden auf dem Client entpackt (`PackAndUnpack.unpackOneFile()`).
5. Drückt der Benutzer den Markieren-Button des Applets, wird *makelist.exe* ausgeführt (`executeCommand(makelist audiofile.mp2 audiofile.lst)`). Dabei müssen sich *makelist.exe* und *bitsplus.txt* im gleichen Verzeichnis wie die MP2-Datei (*audiofile.mp2*) befinden (aufgrund der verwendeten Implementierung des Markierungsalgorithmus des Skalenfaktoren-Audiowasserzeichens).
6. Das Ergebnis der Ausführung von *makelist.exe* ist eine Skalenfaktoren-Datei (*audiofile.lst*), die die herausgesuchten Medienteile darstellt.
7. *Audiofile.lst* wird als ZIP-Datei gepackt (`PackAndUnpack.pack()`).
8. Die ZIP-Datei mit dem Inhalt *audiofile.lst* wird über eine `HttpsURLConnection` zum Server transportiert (`sendPerURLConnection()`). Gleichzeitig wird dem Server die Wasserzeichennachricht mitgeteilt.

Markieren mit dem Markierungsalgorithmus (HttpServlet auf Server)

9. Der Server nimmt die ZIP-Datei mit dem Inhalt *audiofile.lst* vom Client entgegen (`BufferedInputStream inputFromApplet = new BufferedInputStream (request.getInputStream())`).
10. *Audiofile.lst* wird auf dem Server entpackt (`PackAndUnpack.unpackOneFile()`).
11. *Audiofile.lst* wird durch die Ausführung von *wminstxt.exe* mit der übergebenen Wasserzeichennachricht markiert (`executeCommand(wminstxt audiofile.lst audiofilew1.lst 2 0 −1 −2 −1 −1 0 1 3 5 5 3 MESSAGE)`). Das Ergebnis ist eine markierte Skalenfaktoren-Datei (*audiofilew1.lst*).

12. Die markierte Skalenfaktoren-Datei (*audiofilew1.lst*) wird als ZIP-Datei gepackt (`PackAndUnpack.pack()`).

Zusammenfügen markierter Medienteile mit Originalmedium (Applet auf Client)

13. Das Applet lädt die ZIP-Datei, in der sich die markierte Skalenfaktoren-Datei (*audiofilew1.lst*) befindet, vom Server herunter (`download()`).
14. Das Applet lädt die Datei *link.zip* vom Server (`download()`). In *link.zip* befindet sich *link.exe*.
15. Die markierte Skalenfaktoren-Datei (*audiofilew1.lst*) und *link.exe* werden auf dem Client entpackt (`PackAndUnpack.unpackOneFile()`).
16. *Audiofilew1.lst* wird durch die Ausführung von *link.exe* in die originale MP2-Datei (*audiofile.mp2*) eingebracht (`executeCommand(link audiofile.mp2 audiofilew1.lst audiofilew1.mp2)`). Als Ergebnis liegt eine markierte MP2-Datei (*audiofilew1.mp2*) vor.

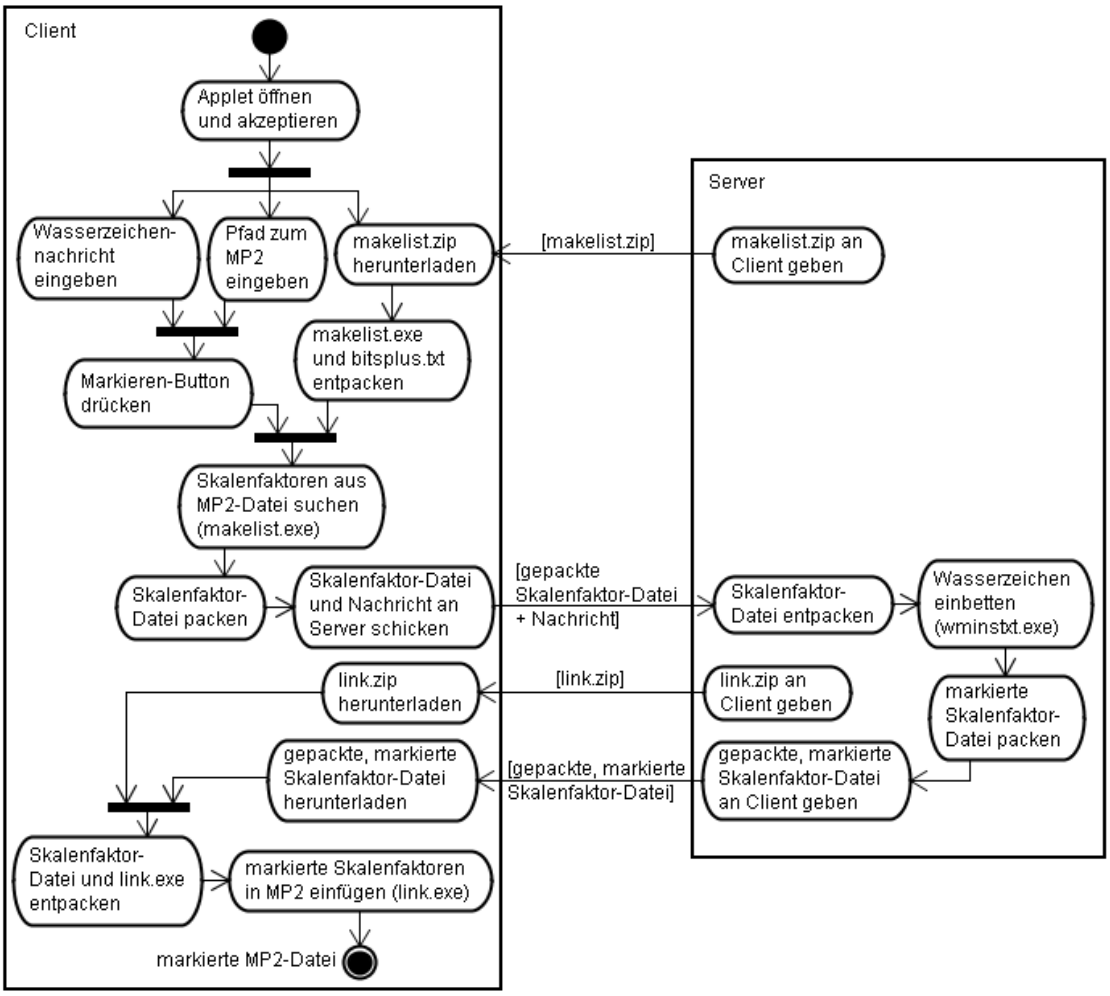

Abbildung 11: Detaillierter Markierungsprozess (Skalenfaktoren-Audiowasserzeichen)

6.2.3 Ausleseprozess

Der Ausleseprozess eines Wasserzeichens beginnt wie der Markierungsprozess mit der *Analyse des Trägermediums und dem Heraussuchen der Medienteile, in denen das Wasserzeichen potentiell enthalten sein kann*. Die herausgesuchten Medienteile werden an den Server gesendet, damit dort das Wasserzeichen ausgelesen werden kann (*Auslesealgorithmus*). Die einzelnen Schritte des Ausleseprozesses sind im Folgenden detailliert aufgelistet und in Abbildung 12 zusammenfassend dargestellt.

Analyse und Heraussuchen (Applet auf Client, wie beim Markierungsprozess)

1. Das Applet wird geöffnet und muss vom Benutzer akzeptiert werden, damit es die nötigen Rechte zum Schreiben und Ausführen hat.
2. Das Applet erwartet vom Benutzer die Eingabe des Pfades zu einer markierten MP2-Datei (*PfadZurDatei/audiofilew1.mp2*).
3. Das Applet lädt die Datei *makelist.zip* vom Server herunter (`download()`). In *makelist.zip* befinden sich *makelist.exe* und *bitsplus.txt*.
4. *Makelist.exe* und *bitsplus.txt* werden auf dem Client entpackt (`PackAndUnpack.unpackOneFile()`).
5. Drückt der Benutzer den Auslese-Button des Applets, wird *makelist.exe* ausgeführt (`executeCommand(makelist audiofilew1.mp2 audiofilew1.lst)`). Dabei müssen sich *makelist.exe* und *bitsplus.txt* im gleichen Verzeichnis wie die markierte MP2-Datei (*audiofilew1.mp2*) befinden.
6. Das Ergebnis der Ausführung von *makelist.exe* ist eine Skalenfaktoren-Datei (*audiofilew1.lst*), die die Medienteile darstellt, in denen das Wasserzeichen potentiell enthalten sein kann.
7. *Audiofilew1.lst* wird als ZIP-Datei gepackt (`PackAndUnpack.pack()`).
8. Die ZIP-Datei mit dem Inhalt *audiofilew1.lst* wird über eine `HttpsURLConnection` zum Server transportiert (`sendPerURLConnection()`).

Wasserzeichen auslesen mit dem Auslesealgorithmus (HttpServlet auf Server)

9. Der Server nimmt die ZIP-Datei mit dem Inhalt *audiofilew1.lst* vom Client entgegen (`BufferedInputStream inputFromApplet = new BufferedInputStream(request.getInputStream())`).
10. *Audiofilew1.lst* wird auf dem Server entpackt (`PackAndUnpack.unpackOneFile()`).
11. *Wmdet.exe* wird ausgeführt (`executeCommand(wmdet audiofilew1.lst 2 0 -1 -2 -1 -1 0 1 3 5)`) und dadurch versucht, das Wasserzeichen aus *audiofilew1.lst* auszulesen. Wenn ein Wasserzeichen vorhanden ist und dieses nicht zerstört wurde, dann kann die Wasserzeichennachricht ausgelesen werden (ansonsten ist keine Nachricht zu finden).
12. Die Wasserzeichennachricht wird an den Client gesendet.

Anzeigen der Wasserzeichennachricht (Applet auf Client)

13. Die Wasserzeichennachricht wird dem Benutzer angezeigt.

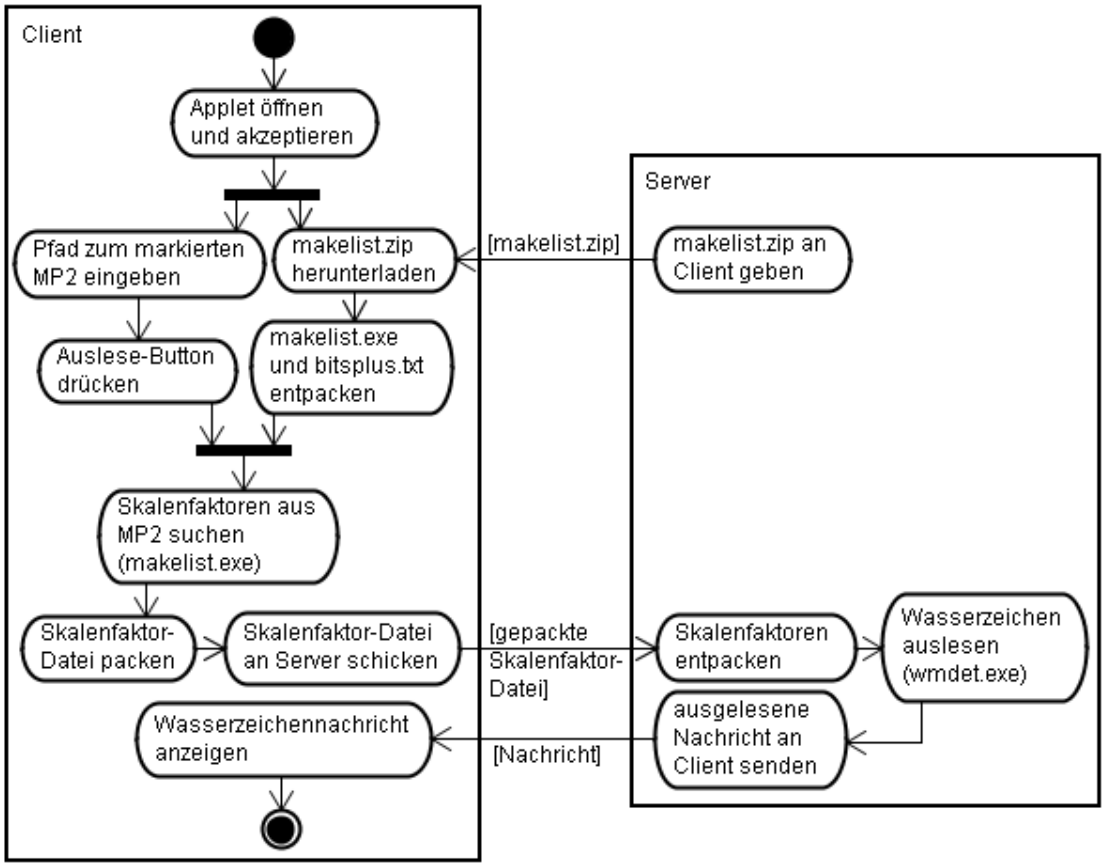

Abbildung 12: Detaillierter Ausleseprozess (Skalenfaktoren-Audiowasserzeichen)

6.3 Diskussion

In dieser Diskussion werden die Ziele Sicherheit der Wasserzeichen, Sicherheit der Originalmedien sowie Sicherheit und Performanz des Systems anhand der in Abschnitt 5.2 gewünschten Ziele und den dort aufgestellten Bedingungen untersucht. Dabei interessiert, ob die Bedingungen eingehalten werden konnten und die erwünschten Ziele erreicht worden sind. Des Weiteren werden Vor- und Nachteile der konkreten Implementierung des Skalenfaktoren-Audiowasserzeichens dargelegt und schließlich mögliche Anwendungsgebiete eines solchen Systems vorgestellt.

6.3.1 Ziele und Bedingungen

In dieser Arbeit ist ein Client-Server-System im Internet mit webbasierter Kommunikation umgesetzt. Dieses soll von jedem und überall verwendet werden können und den Einsatz digitaler Wasserzeichen flexibel gestalten. Dadurch ist vielen Benutzern die Anwendung digitaler Wasserzeichen möglich, wobei sie ihre Medien damit markieren und so kenntlich machen können. Die Wasserzeichenentwickler hingegen haben die Absicht, ihre Wasserzeichen Kunden anzubieten und sie so zu verbreiten. Weiterhin möchten sie ihre Wasserzeichenverfahren kommerziell zur Finanzierung der Entwicklungsarbeiten nutzen.

Rechenlast des Servers
Es ist erwünscht, dass die Kosten zur Bereitstellung eines Client-Server-Systems so gering wie möglich gehalten werden. Dies ist am ehesten durch möglichst niedrige Serverkosten erreichbar, wofür es erforderlich ist, dass der Server so wenige Aufgaben wie möglich hat, also der Client dem Server soviel Arbeit wie möglich abnimmt. Dafür sollte das Wasserzeichenverfahren in Komponenten geteilt werden, sodass der Server nur den eigentlichen Markierungsalgorithmus ausführt und die verbleibenden Aufgaben auf dem Client stattfinden können. Kann ein Wasserzeichenverfahren in die Komponenten aus Abschnitt 4.3 aufgeteilt werden, so ist die genannte Vorgehensweise realisierbar, wie dies beispielhaft am Skalenfaktoren-Audiowasserzeichen gezeigt wurde. Somit wird die Rechenlast des Servers gering gehalten, wodurch die Kosten niedrig bleiben. Dies darf jedoch nicht die Nutzung des Systems beeinträchtigen, was bedeutet, dass es mehreren Kunden gleichzeitig möglich sein muss darauf zuzugreifen. Da hier eine Forschungsarbeit angestellt wurde und zunächst lediglich eine beispielhafte Umsetzung erfolgt ist, kann dieses Ziel nicht nachgewiesen werden. Es ist jedoch offensichtlich, dass durch die Verringerung der Rechenlast des Servers dessen Bereitstellungskosten vermindert sind.

Damit der Client dem Server Aufgaben abnehmen kann, benötigt er eine aktive Komponente. In diesem Projekt wurde die Programmiersprache Java verwendet und auf der Clientseite ein Applet implementiert, durch das der Client seine zugewiesenen Aufgaben übernehmen kann.

Sicherheit der Wasserzeichenalgorithmen
Durch die Grundvoraussetzung und Motivation der Arbeit, dass der Wasserzeichenalgorithmus auf dem Server verbleiben muss, sollen effizientere Angriffe auf Wasserzeichen vermieden werden. Dies ist bei der Implementierung des Skalenfaktoren-Wasserzeichens erfüllt.

Es ist jedoch nicht bei jedem Wasserzeichenverfahren möglich, es in die Komponenten aus Abschnitt 4.3 aufzuteilen, wie am Beispiel des Zhao-Koch-Wasserzeichens in Abschnitt 4.4.4 deutlich wird. Solche Wasserzeichenverfahren müssen entweder angepasst werden, sodass sie in die geforderten Komponenten gegliedert werden können, oder sie müssen komplett auf dem Client oder dem Server ausgeführt werden. Die gesamte Ausführung auf dem Server vorzunehmen hat sich aus Performanzgründen als nicht praktikabel herausgestellt, wogegen die gesamte Ausführung auf dem Client der Bedingung widerspricht, dass der Wasserzeichenalgorithmus auf dem Server verbleiben muss.

Sicherheit der Originalmedien
Der Client hat die Aufgabe, Teile aus dem Medium herauszusuchen und nur diese an den Server zu übertragen. Dadurch, dass der Client seine Medien nicht komplett an den Server überträgt, ist die Sicherheit der Originalmedien gewährleistet. In der konkreten Umsetzung werden Skalenfaktoren aus einem MP2-Audio herausgesucht und nur diese an den Server gesendet. Die Übertragung der herausgesuchten Skalenfaktoren vom Client an den Server erfüllt zwar das Ziel der Sicherheit der Originalmedien, jedoch nicht ohne weitere Maßnahmen das Ziel der bestmöglichen Performanz.

Performanz des Systems
Der Größenvorteil der Skalenfaktoren im Gegensatz zum gesamten Audio liegt bei etwa drei bis zehn Prozent, denn soviel stellen die Skalenfaktoren bezüglich des gesamten Audiostroms dar. Diese drei bis zehn Prozent können jedoch nicht als absoluter Größengewinn gerechnet werden, denn aufgrund der prototypischen Implementierung des Wasserzeichenverfahrens ist die durch die Ausführung von *makelist.exe* erstellte Skalenfaktoren-Datei um ein vielfaches Größer als das Originalaudio. Der Grund dafür kann in [Steinebach99] nachgelesen werden. Würde diese Skalenfaktoren-Datei an den Server übertragen werden, so wäre statt einem Gewinn in der zu übertragenden Größe das Gegenteil bewirkt. Wird die Skalenfaktoren-Datei als ZIP-Datei komprimiert, so stellt sie nur noch etwa ein Fünftel des ursprünglichen Audios dar. Dieser Wert stellt gleichzeitig den gesamt zu betrachtenden Größenvorteil dar, mit dem die herausgesuchten Medienteile an den Server gesendet werden können. Darüber hinaus muss die Zeit zum Packen und Entpacken in die Performanzrechnung mit einbezogen werden. Diese ist allerdings verhältnismäßig kurz, sodass durch das Komprimieren kaum Performanzeinbußen verursacht werden.

Um eine bewertende Aussage bezüglich der Performanz treffen zu können, muss neben der Minimierung der Performanzeinbußen durch die zu übertragende Dateigröße auch die Performanz des gesamten Systems betrachtet werden. Tests haben gezeigt, dass das Markieren und das Auslesen durch die Ausführung des umgesetzten Client-Server-Modells nur geringfügige Performanzeinbußen gegenüber der lokalen Ausführung der Executables aufweisen. Dabei wird die Ausführungsdauer um etwa zehn Prozent verlängert. Das Ziel der bestmöglichen Performanz deckt ein gemeinsames Interesse von Benutzer und Wasserzeichenentwickler ab und kann mit den erreichten zehn Prozent folglich als erreicht angesehen werden.

Sicherheit des Systems
Das gewünschte Ziel der Sicherheit des Systems kann nur erreicht werden, wenn das gesamte System sicher ist und wird durch eine einzige Sicherheitslücke gefährdet. Das bedeutet, dass Client, Server und das Verbindungsnetz sicher sein müssen. Die Sicherheit der Verbindung ist hier durch HTTPS abgedeckt, wodurch eine vertrauliche Datenübertragung sowie die Authentizität des Servers versichert sind. Die Sicherheit von Client und Server in Form von Firewalls oder Ähnlichem ist in vorliegender Arbeit als gegeben betrachtet. Zusätzlich wird ein signiertes Applet eingesetzt, das die Authentizität, Verbindlichkeit und Integrität des heruntergeladenen Applet-Codes gewährleistet. Diese Sicherheitsvorkehrungen sind für die vorliegende Arbeit ausreichend.

Vor- und Nachteile der Implementierung des Skalenfaktoren-Audiowasserzeichens
Der große Vorteil, den das Skalenfaktoren-Audiowasserzeichen bei der Implementierung in einem Client-Server-Modell bietet, besteht darin, dass es bereits in geeignete Komponenten geteilt vorliegt (siehe Abschnitt 4.4.1). Dadurch können einfach Schritte unterschieden werden und Client und Server leicht ihre Aufgaben zugeteilt bekommen, ohne dabei das Verfahren an sich verändern zu müssen. Aus diesem Grund eignet sich dieses Wasserzeichenverfahren besonders für das hier entworfene und umgesetzte Modell.

Bei der Implementierung des Skalenfaktoren-Audiowasserzeichens ergibt sich der Nachteil, dass es bei der Ausführung der Executables nicht möglich ist, Pfadangaben zu einem Medium anzugeben, sondern es muss die Executable in dem Verzeichnis ausgeführt werden, in dem sich das angegebene Medium befindet. Das bedeutet für die konkrete Umsetzung, dass Applet und Servlet diesem Problem entgegenwirken müssen, indem die zu markierenden Medien sowie die erstellten Skalenfaktoren-Dateien mit den Executables in einem Verzeichnis zusammengebracht werden.

Ein weiterer Nachteil resultiert aus der Plattformabhängigkeit der Executables: Das gesamte Client-Server-System ist nicht plattformunabhängig, obwohl es über das Internet verwendet werden kann und mit Java als plattformunabhängige Sprache umgesetzt wurde.

6.3.2 Anwendungsgebiete eines solchen Systems

Ein Teil dieser Arbeit ist die beispielhafte Umsetzung des Skalenfaktoren-Audiowasserzeichens, wodurch gezeigt wurde, dass das entwickelte Modell realisierbar ist. Andere Wasserzeichenverfahren müssen vor einem Einsatz in einem solchen System zuerst analysiert werden, denn jedes Wasserzeichen ist anders implementiert und hat beim Einsatz in einem Client-Server-System andere Voraussetzungen. Wenn ein Wasserzeichenverfahren nicht in die geforderten Komponenten gegliedert oder diesen angepasst werden kann, so besteht ein Konflikt zwischen der Sicherheit der Wasserzeichen (siehe Abschnitt 5.8.4), der Performanz des gesamten Ablaufs (siehe Abschnitt 5.8.5) und der Eignung des Wasserzeichenverfahrens für dieses Modell. Andererseits kann bei der Markierung von kleinen Medien gegebenenfalls ein Transfer des gesamten Mediums an den Server akzeptabel sein. Die Schutzziele müssen für jede Umsetzung untersucht und je nach Anwendung gewichtet werden. Bei Betrachtung der genannten Bedingungen und Ziele muss für jedes Wasserzeichenverfahren individuell erfasst werden, welche Ziele mit einem Client-Server-Modell erreicht werden sollen und abgewogen werden, ob sich das Verfahren eignet und wie eine konkrete Umsetzung aussehen könnte bzw. ob sie sich rentieren würde.

Mögliche Anwendungsgebiete des Systems

Trotz Erfüllung aller Ziele stellt sich die grundlegende Frage nach möglichen Anwendungsgebieten eines *Client-Server-Systems zum sicheren Markieren von Medien mit digitalen Wasserzeichen*. Das Client-Server-System bietet, neben dem flexiblen Einsatz digitaler Wasserzeichen, ein variables Zusammentreffen der Benutzer (Kunden) mit den Wasserzeichenentwicklern (Anbietern). Die Sicherheit steht dabei in vielerlei Hinsicht im Vordergrund. Für ein sicheres Markieren ist neben der Sicherheit, die das System bieten muss, die Sicherheit der Wasserzeichen von Interesse. Wasserzeichenentwickler beabsichtigen ihre Algorithmen nicht an die Kunden herauszugeben, um sie nicht durch effizientere Angriffe zu gefährden. Die Kunden müssen sich auf die Sicherheit ihrer Medien verlassen können, die sie durch deren Markieren erreichen und nicht durch ein unsicheres System bzw. durch das Herausgeben ihrer Originalmedien an den Server gefährden wollen.

Sicherheit der Wasserzeichen durch Authentifizierung des Clients
Ist ein Wasserzeichenverfahren durch den Einsatz in dem hier entwickelten System nun sicherer? Um diese Frage beantworten zu können, wird folgende Überlegung angestellt: Was passiert, wenn jeder Benutzer ohne Authentifizierung das hier entwickelte System nutzen kann? Dann kann aus einem markierten Medium, das in „fremde“ Hände gelangt ist, die Wasserzeichennachricht unautorisiert ausgelesen werden. Des Weiteren kann das Medium beliebig manipuliert werden, bis das Wasserzeichen nicht mehr auszulesen, also zerstört, ist. Damit ist die Sicherheit des Mediums nicht mehr gewährleistet und es tritt ein Sicherheitsrisiko auf. Damit unautorisierte Ausleseversuche unterbunden werden können, müsste der geheime Schlüssel individuell für jeden Benutzer oder sogar jedes Medium festgelegt werden. Die individuelle Festlegung des geheimen Schlüssels kann in Kombination einer Authentifizierung des Benutzers realisiert werden, wobei der geheime Schlüssel entweder in Zusammenhang mit der eindeutigen Identität des Benutzers steht oder vom Benutzer selbst festgelegt wird.

Kann der Schlüssel vom Benutzer bei jeder Markierung erneut gewählt werden, so entstehen beim Markieren eines Mediums jedes Mal verschiedene markierte Medien. Dadurch ist der Benutzer im Besitz mehrerer markierter Medien, die sich nur in ihrem Schlüssel unterscheiden und durch deren Vergleich der Benutzer Rückschlüsse auf den Zusammenhang zwischen Wasserzeichen und geheimen Schlüssel ziehen könnte. Dies kann ein Sicherheitsrisiko darstellen und sollte vermieden werden, indem der geheime Schlüssel von der Identität des Benutzers abhängig gemacht wird.

Zur Authentifizierung des Clients kann das hier entworfene Modell anstatt eine separate Anwendung zu bilden, in eine bereits bestehende integriert werden, zum Beispiel in das Fraunhofer Watermarking-Portal [Watermarking]. Dort ist die Authentifizierung des Benutzers umgesetzt und es wird eine eindeutige Identität für jeden Benutzer als geheimer Schlüssel verwendet. Die Kenntnis dieses Schlüssels besitzt nur der Server und nicht der Client.

Sind in einem Client-Server-System die Bedingungen individueller geheimer Schlüssel und Authentifizierung der Benutzer zusätzlich zu den durch das hier entworfene Modell aufgestellten Bedingungen erfüllt, so sind die Wasserzeichenverfahren sicherer. Es soll lediglich zum Bedenken gegeben werden, dass ein verteiltes System einen größeren Angriffspunkt darstellt als ein einzelner Rechner und so auch gegebenenfalls eher eine Lücke offen lässt, durch die die gesamte Sicherheit des Systems und somit auch der Wasserzeichen gefährdet werden kann.

Der Einsatz digitaler Wasserzeichen in einem Client-Server-System gewährleistet die Sicherheit der Medien. Diese ist bei kompletter Ausführung beim Anbieter nicht garantiert, denn dabei muss das Originalmedium komplett zum Markieren herausgegeben werden.

Sinnvoller Einsatz des Systems
Damit ist diese Diskussion zu den Bedingungen gelangt, unter denen ein solches System sinnvoll eingesetzt werden kann. Dazu zählen die Sicherheit der Wasserzeichen und des Systems, sowie die Authentifikation beider Kommunikationspartner. Die Sicherheit der Wasserzeichen ist dadurch gewährleistet, dass der Markierungsalgorithmus nicht vom Server herausgegeben wird und der geheime Schlüssel individuell für jeden Benutzer oder jedes Medium vergeben wird. Das System muss gängigen Sicherheitsmaßnahmen entsprechen und die Authentifikation der Kommunikationspartner garantieren. Diese

Bedingungen müssen nicht für jede Implementierung erneut umgesetzt werden, sondern können auch dadurch erreicht werden, dass das hier entworfene Client-Server-Modell in andere Anwendungen integriert wird. Eine mögliche Zusammenführung mit dem Watermarking-Portal ist bereits angesprochen worden. Ansonsten könnte ein Online-Shop eine mögliche Anwendung darstellen, bei dem Kunden gegen Bezahlung das Markieren ihrer eigenen Medien mit digitalen Wasserzeichen angeboten wird.

Universeller Einsatz des Systems
Abschließend die Frage, ob dieses Modell universell einsetzbar ist. Dafür wird die Aufteilung digitaler Wasserzeichenverfahren in die allgemeinen Komponenten aus Abschnitt 4.3 weiter abstrahiert, sodass lediglich sicherheitsrelevante und –irrelevante Komponenten unterschieden werden. Dabei stellen die Schritte *Markierungsalgorithmus* sicherheitsrelevante und *Analyse des Trägermediums und Heraussuchen der zum Markieren relevanten Medienteile* sowie *Zusammenfügen markierter Medienteile mit Originalmedium* sicherheitsirrelevante Komponenten dar. Die sicherheitsirrelevanten Schritte, zu denen beispielsweise ein Parsen der Medien oder Formatwandlungen zu zählen sind, können auf dem Client ausgeführt werden, um so dem Server Arbeit abzunehmen. Die als sicherheitskritisch angesehenen Verfahrensteile, zu denen insbesondere der Markierungsalgorithmus gehört, müssen auf dem Server ausgeführt werden um ausreichend Schutz vor Angriffen bieten zu können.

Aus dieser Erkenntnis, dass Wasserzeichenverfahren abstrakt betrachtet in sicherheitsrelevante und –irrelevante Komponenten unterteilt werden können, folgt, dass Wasserzeichenentwickler sich bereits bei Neuentwicklungen Gedanken darüber machen können, welche Komponenten ohne Risiko auf dem Client ausgeführt werden könnten. Wird der Server nicht unnötig mit sicherheitsirrelevanten Aufgaben belastet, kann dadurch die Performanz des Systems gesteigert werden. Außerdem können Schwierigkeiten bei der Umsetzung eines konkreten Wasserzeichenverfahrens in einem Client-Server-Modell vermieden werden, wenn bereits bei der Implementierung eine Teilung in Komponenten im Hinblick auf spätere Aufgaben für Client und Server und unter Berücksichtigung von deren Sicherheitsrelevanz beachtet wird.

Kapitel 7

Zusammenfassung

Zum Abschluss wird zunächst eine kurze Zusammenfassung dieser Arbeit gegeben. Schließlich werden in Abschnitt 7.1 ein Fazit der gewonnenen Kenntnisse und des entworfenen Modells und in Abschnitt 7.2 ein Ausblick auf mögliche Fortführungen zu diesem Projekt gegeben.

Zu Beginn dieser Arbeit wird beschrieben, warum ein Client-Server-Modell zum sicheren Markieren von Medien mit digitalen Wasserzeichen untersucht und entworfen werden soll. Dafür werden zunächst notwendige Grundlagen zum Verstehen der Arbeit gelegt. Bei einem solchen Client-Server-Modell stehen immer zwei Interessensparteien in Verbindung, die Benutzer auf der Clientseite mit den Wasserzeichenentwicklern auf der Serverseite.

Die Interessen der Benutzer und Wasserzeichenentwickler werden für das zu entwickelnde Client-Server-Modell herausgearbeitet und daraus Bedingungen an das Modell abgeleitet. Außerdem werden Wasserzeichenverfahren im Hinblick auf eine Aufteilung in Komponenten analysiert und die allgemein herausgearbeitete Aufteilung mit konkreten bestehenden Verfahren verglichen. Wasserzeichenverfahren bestehen meist aus folgenden Komponenten: Analysieren der Medien, Heraussuchen der zum Markieren notwendigen Medienteile, Markierungsalgorithmus und Zusammenfügen der markierten Medienteile mit dem Originalmedium.

Im Entwurf werden die herausgearbeiteten Komponenten der Wasserzeichenverfahren als Aufgaben an Client und Server verteilt und die Anforderungen an die Client- und an die Serverkomponente und ebenfalls an deren Zusammenspiel herausgearbeitet. Client und Server müssen zusammen eine Aufgabe erfüllen und dafür miteinander kommunizieren. Dabei muss unter anderem die Sicherheit des Transports (HTTPS) gewährleistet werden, um die Sicherheit des gesamten Client-Server-Systems erzielen zu können. Des Weiteren ist die Performanz des Systems von Interesse. Als Ergebnis des Entwurfs liegt schließlich ein Client-Server-Modell vor.

Bei der Umsetzung des entworfenen Modells mit den aufgestellten Anforderungen werden der Sicherheit und Performanz besondere Aufmerksamkeit gewidmet. Beispielhaft ist das Skalenfaktoren-Audiowasserzeichen implementiert und die Umsetzung diskutiert.

7.1 Fazit

Jedes Wasserzeichen hat andere Anforderungen, benötigt andere Parameter, ist individuell implementiert und kann in andere Komponenten aufgeteilt werden. Aus dieser Erkenntnis folgt, dass jedes Wasserzeichen in einem Client-Server-Modell, wie es in dieser Arbeit vorgestellt wurde, andere Anforderungen an die konkrete Umsetzung hat. Es gibt Wasserzeichenverfahren, die sich für die Umsetzung in einem solchen Modell besser eignen als andere. Das hängt unter anderem mit der Aufteilung ihrer Komponenten zusammen und damit, wie viele Aufgaben schließlich auf den Client ausgelagert werden können. Es gibt auch Wasserzeichenverfahren, bei denen eine Umsetzung in einem solchen Modell überhaupt nicht realisierbar ist, weil sich das Wasserzeichenverfahren beispielsweise nicht sinnvoll in Komponenten aufteilen lässt. Bei anderen Wasserzeichenverfahren, bei denen die Ausführung des Markierungsalgorithmus nicht auf dem Server möglich ist, kann deren Sicherheit zu stark gefährdet sein, sodass es sehr fraglich ist, ob eine Umsetzung in einem Client-Server-Modell einen Mehrwert bringen würde.

Neben der Sicherheit muss die Performanz des gesamten Systems berücksichtigt werden. Beispielsweise kann es bei Bildwasserzeichen einen erheblichen Mehraufwand bedeuten, eine Komponente des Wasserzeichenverfahrens extra auf den Client zu laden, um sie dort auszuführen. Es kann durchaus vorkommen, dass ein Medium klein genug ist, sodass es performanter als Ganzes auf den Server übertragen und dort markiert werden kann, anstatt es zuvor auf dem Client zu analysieren und zu zerteilen. Denn bei einem kleinen Medium sind die Größe für die Übertragung und der Speicherbedarf auf dem Server verhältnismäßig gering. Dies würde zwar die Sicherheit der Originalmedien auf die Art und Weise beeinträchtigen, dass der Server das Originalmedium besitzt. Jedoch kann es durchaus sein, dass ein Benutzer dem Server vertraut und deswegen keine Bedenken hat, seine Originalmedien an diesen herauszugeben. Der Unsicherheit, die während der Übertragung auftritt, kann durch die Verwendung von SSL vorgebeugt werden.

Es muss bei jeder Implementierung eines hier vorgestellten Client-Server-Systems individuell ein geeigneter Kompromiss zwischen den Interessen von Benutzer und Wasserzeichenentwickler im Hinblick auf die Performanz des Systems und die Sicherheit der Wasserzeichen sowie der Originalmedien gefunden werden. Dafür müssen unter anderem

- das Vertrauensverhältnis zwischen Benutzer und Serverbetreiber bzw. das Vertrauen des Benutzers dem Serverbetreiber gegenüber,
- die Performanz der Übertragung und der Speicherbedarf auf dem Server, die in direktem Zusammenhang mit der Art und Größe des Mediums steht,
- die Eignung des Wasserzeichens zur Aufteilung in Komponenten und
- die Sicherheit des Wasserzeichens im Hinblick auf effizientere Angriffe

betrachtet werden.

Weitere Aspekte, die bei der Umsetzung eines konkreten Wasserzeichenverfahrens in einem Client-Server-System von Bedeutung sein können, sind

- die Leistung und Speichergröße, die der Server bereitstellt,
- die Absichten der Benutzer im Hinblick auf Verwendung der Wasserzeichen, sowie
- die Bandbreite von deren Internetverbindungen,
- die geschätzte Anzahl gleichzeitiger Zugriffe auf das System und
- die Anzahl der bereitgestellten Wasserzeichenverfahren in einem System.

Bei einer Implementierung von einem konkreten Wasserzeichenverfahren in einem hier entwickelten Client-Server-Modell muss jedes erwünschte Ziel und die Eignung der Wasserzeichenverfahren individuell erfasst werden. Dafür muss jede Bedingung individuell gewichtet und dementsprechend stark berücksichtigt werden und es muss abgewogen werden, ob sich das Wasserzeichenverfahren eignet und wie eine konkrete Umsetzung aussehen könnte bzw. ob sie sich rentieren würde.

Sind in einem hier entwickelten Client-Server-System die Bedingungen Benutzerauthentifizierung und individueller geheimer Schlüssel erfüllt, so sind die Wasserzeichenverfahren sicher. Dabei bietet das Client-Server-System vielfältige Möglichkeiten für ein variables Zusammentreffen der Wasserzeichenentwickler und Benutzer sowie einen flexiblen Einsatz digitaler Wasserzeichen. Die Sicherheit ist stets im Vordergrund, damit ein sicheres Markieren möglich ist. Dafür verbleiben die Wasserzeichenalgorithmen sicher auf dem Server, um nicht durch effizientere Angriffe gefährdet zu werden. Die Kunden müssen ihre Originalmedien nicht komplett herausgeben und können sich so auf deren Sicherheit verlassen.

7.2 Ausblick

In diesem Abschnitt werden Herausforderungen an fortführende Arbeiten zu diesem Thema gegeben. Dabei kann allgemein versucht werden, die Performanz des gesamten Ablaufs zu verbessern durch eine Art Dateimanagement in dem Sinn, dass der Client vor dem Herunterladen von Komponenten des Wasserzeichenverfahrens nachschaut, ob er diese bereits heruntergeladen hat. Wenn die benötigte Komponente auf dem Client vorliegt, beispielsweise, weil das Wasserzeichenverfahren bereits verwendet wurde, so muss der Client die Komponente nicht erneut vom Server laden und erspart so einen unnötigen Download. Beim Skalenfaktoren-Audiowasserzeichen ist diese Überlegung wahrscheinlich nicht notwendig, da die verwendeten Executables von sehr geringer Größe sind, erst recht, wenn sie mit den Audiodaten verglichen werden.

Konkurrierend zu der Überlegung der mehrfachen Nutzung von heruntergeladenen Dateien steht das „Saubermachen“ nach der Nutzung des Client-Server-Modells. Dabei könnten nach Abschluss eines Markierungs- oder Ausleseprozesses die heruntergeladenen Komponenten des Wasserzeichenverfahrens und auch die herausgesuchten Medienteile, hier die Skalenfaktoren-Datei, auf dem Client gelöscht werden. Die nicht mehr verwendeten Dateien werden also sowohl auf dem Client, als auch auf dem Server gelöscht.

Die konkrete Umsetzung des Skalenfaktoren-Audiowasserzeichens könnte durch das Einstellen wasserzeichenspezifischer Parameter erweitert werden. Momentan kann lediglich die Wasserzeichennachricht vom Benutzer selbst eingeben werden. Alle anderen Parameter, wie etwa der geheime Schlüssel, sind fest eingestellt. Es kann für andere Implementierungen interessant sein, wasserzeichenspezifische Parameter wie zum Beispiel die Kapazität individuell einstellen zu können. Besonders beim Parameter des geheimen Schlüssels kann die Unterscheidung, ob der Benutzer oder nur der Server den Schlüssel kennt, relevant sein.

Zum Integrieren der Benutzerauthentifizierung kann eine Integration in bereits bestehende Anwendungen wie das Watermarking-Portal vorgenommen werden. Des Weiteren kann der Einsatz eines hier entwickelten Systems auch in einem Online-Shop von Interesse sein. Bei der Betrachtung eines universellen Einsatzes des Systems können von vorneherein die Aufgaben von Client und Server herausgearbeitet werden, indem das neu zu entwickelnde Wasserzeichenverfahren von vorneherein in sicherheitsrelevante und –irrelevante Komponenten geteilt wird.

Eine ganz andere Art der Weiterentwicklung dieses Client-Server-Systems wendet sich von den Wasserzeichen weg hin zu partiellen Verschlüsselungsverfahren. Vielleicht wäre neben dem Einsatz von Wasserzeichenverfahren auch der Einsatz partieller Verschlüsselungsverfahren in einem Client-Server-System, ähnlich dem hier entwickelten, überlegenswert. Dabei ist die Motivation der Algorithmensicherheit nebensächlich, denn Verschlüsselungsverfahren entsprechen normalerweise dem Kerckhoffs-Prinzip. Das bedeutet, dass sie veröffentlicht werden können, ohne effizientere Angriffe befürchten zu müssen. Statt der Algorithmensicherheit ständen finanzielle Aspekte, die etwa eine zeitliche Bezahlung für die Nutzung der Verschlüsselungsverfahren erlauben, im Vordergrund. Es käme auch hierbei immer auf die gewünschten Motivationsschwerpunkte und die Gewichtung der Interessen an. Diese sollen jedoch nicht weiter ausgeführt werden, da sie eine andere Arbeit darstellen und nicht dem Ziel dieser entsprechen.

Quellen

[AdKr03] Adams, M., Krause, M.: *Java Security Architecture*. Fachhochschule Bonn-Rhein-Sieg, http://www2.inf.fh-bonn-rhein-sieg.de/~rberre2m /lehre/ss03/vups1/Seminar/4_JavaSecurityArchitecture.pdf, Juni 2003

[Bengel04] Bengel, G.: *Grundkurs Verteilte Systeme*. Vieweg Verlag, Wiesbaden, ISBN: 3-528-25738-5, 3. Auflage, 2004

[CaFoFu05] Cayre, F., Fontaine, C., Furon, T.: *Watermarking Security Part One: Theory*. Proceedings of SPIE, Volume 5681, pp. 746-757, March 2005

[CoMiBl02] Cox, I. J., Miller, M. L., Bloom, J. A.: *Digital Watermarking*. Academic Press, San Diego, ISBN: 1-55860-714-5, 2002

[Dittmann00] Dittmann, J.: *Digitale Wasserzeichen*. Springer Verlag, Berlin, Heidelberg, ISBN: 3-540-66661-3, 2000

[Eckert05] Eckert, C.: *IT-Sicherheit – Konzepte – Verfahren – Protokolle*. Oldenbourg Wissenschaftsverlag, München, ISBN: 3-486-57676-3, Studienausgabe, 2005

[Fitzinger03] Fitzinger, J.: *Java Security*. Seminararbeit, http://www.ssw.uni-linz.ac.at/Teaching/Lectures/Sem/2003/reports/Fitzinger/Fitzinger.pdf, 2003

[ISO93] ISO/IEC 11172: *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3: Audio*. International Standard, Genf, 1. Ausgabe vom 01.08.1993

[ISO94] ISO/IEC 7498-1: *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*. International Standard, Genf, 1. Ausgabe von 1994, 2. Ausgabe vom 15.06.1996

[Kopp98] Kopp, M.: *Javas Sandbox-Modell und signierte Applets*. In iX Magazin für professionelle Informationstechnik, Heise Verlag, http://www.javasolutions.de/artikel/security/sandbox.html, Ausgabe 6, Juni 1998

[Pilz97] Pilz, M.: *Das Sicherheitskonzept von Java*. http://www.ifi.unizh.ch/richter/people/pilz/pubb/Pilz97a.pdf, Institut für Informatik, Universität Zürich, Juni 1997

[Schneider91] Schneider, H.-J. (Hrsg.): *Lexikon der Informatik und Datenverarbeitung*. Oldenbourg Verlag, München, ISBN: 3-486-21514-0, 3. Auflage, 1991

[SiAt04] Sion, R., Atallah, M.: *Attacking Digital Watermarks*. Proceedings of SPIE, Volume 5306, pp. 848-858, January 2004

[Steinebach99] Steinebach, M.: *Copyright Protection für digitales Audio*. Diplomarbeit an der Technischen Universität Darmstadt, Nr.-KOM-D-0068, März 1999

[StZm04] Steinebach, M., Zmudzinski, S.: *Partielle Verschlüssleung von MPEG Audio*. D A CH Security 2004, Syssec – IT Security & IT Management, Patrick Horster (Hrsg.), ISBN: 3-00-013137-X, pp. 470-484, 2004

[Sun] Sun Microsystems, Inc., http://www.sun.com/

[TaSt03] Tanenbaum, A., van Steen, M.: *Verteilte Systeme – Grundlagen und Paradigmen*. Pearson Studium, München, ISBN: 3-8273-7057-4, 2003

[VoPePuEgSu01] Voloshynovskiy, S., Pereira, S., Pun, T., Eggers, J. J., Su, J. K.: *Attacks on Digital Watermarks: Classification, Estimation-Based Attacks, and Menchmarks*. IEEE Communications Magazine, Volume 39, Issue 8, pp. 118-126 , August 2001

[W3C06] World Wide Web Consortium (W3C): *Web Services Activity*. http://www.w3.org/2002/ws/, 2002-2006

[Watermarking] Mediensicherheit in der IT (MERIT) des Fraunhofer IPSI: Fraunhofer Watermarking-Portal. http://watermarkingportal.ipsi.fraunhofer.de

[ZaKo95] Zhao, J., Koch, E.: *Embedding Robust Labels Into Images For Copyright Protection*. In Proc. of the Int. Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Technologies, Vienna, August 1995