



Technisch-Naturwissenschaftliche  
Fakultät

# Sicherheitsaspekte in Applikationsprotokollen

## SEMINAR

LVA-Nummer: 353.016, WS 2011

Eingereicht von:  
Alexander Ritt  
Michal Wasilewski

Angefertigt am:  
Institut für Informationsverarbeitung und Mikroprozessortechnik

Beurteilung:  
DI. Dr. Peter René Dietmüller

Linz, November 2011

---

# *HTTP/HTTPS, WebDAV & CalDav*

---

## **Inhaltsverzeichnis**

---

<b>1. Das Hyper-Text-Transfer-Protokoll (http)</b>	2
<b>1.1. Geschichte von http</b>	2
<b>1.2. http-Methoden</b>	2
1.2.1. HTTP – GET	2
1.2.2. HTTP – PUT	2
1.2.3. HTTP – OPTIONS	2
1.2.4. HTTP – HEAD	3
1.2.5. HTTP – POST	3
1.2.6. HTTP – DELETE	3
1.2.7. HTTP – TRACE	3
1.2.8. HTTP – CONNECT	3
<b>1.3. HTTP Header</b>	3
1.3.1. Request Header	3
1.3.2. Response Header	3
<b>1.4. Kommunikation mittels http</b>	3
1.4.1. Verbindungsaufbau	4
1.4.2. Statuscodes	4
1.4.3. Beispiel	4
<b>1.5. Authentifikation mittels http</b>	5
<b>2. HTTPS (RFC 2818)</b>	7
<b>2.1. SSL Aufbau</b>	7
<b>2.2. SSL Handshake</b>	7
<b>2.3. SSL und Zertifikate</b>	8
<b>3. WebDAV</b>	9
<b>3.1 Geschichte</b>	9
<b>3.2 Funktionsweise</b>	9
<b>3.3 Umsetzung / Anwendungen</b>	10
<b>4. CalDAV</b>	11
<b>4.1 Geschichte</b>	11
<b>4.2 Funktionsweise</b>	11
<b>4.3 Umsetzung / Anwendungen</b>	12
<b>5. Literaturverzeichnis</b>	13

## 1. Das Hyper-Text-Transfer-Protokoll (http)

---

### 1.1. Geschichte von http

---

Das von Tim Berners Lee entwickelt Hyper-Text-Transfer-Protokoll war als einfaches Protokoll für die Übertragung von Daten entwickelt worden. In der ursprünglichen Version 0.9 unterstütze das Protokoll nur einen Teil der heute üblichen Methoden. In der ursprünglichen Version waren auch keine Erweiterungen wie MIME oder ein Authentifizierungsmechanismus vorgesehen. Mit der Version 1.0 (RFC 1945) wurde die Kommunikation mittels Request-Response Verfahren eingeführt, dies ermöglicht eine bidirektionale Übermittlung von Informationen. Die Möglichkeit mit persistenten Verbindungen zu arbeiten war nur eine Neuerung in der Version 1.1 (RFC 2616). Während bei http 1.0 der Client für jede Anfrage eine neue Verbindung zum Server aufbauen musste, können in der Version 1.1 Verbindungen zwischen Client und Server aufrecht erhalten werden und damit können mehrere Anfragen gesendet werden ohne die Verbindung trennen zu müssen. Wenn nun ein Client (z.B. Webbrowser) mehrere Anfragen (Requests) an den Server sendet ohne zuvor auf die Antwort (Response) der vorhergehenden Anfrage zu warten, so nennt man diese Funktion in der Version 1.1 Request-Pipelining. Die Funktionalität der persistenten Verbindungen, des Request-Pipelining und die Cache-Kontrollfunktionen brachten einen deutlichen Performancegewinn.

Der einfache Aufbau des Protokolls, die dadurch vergleichsweise einfache Implementierung des Protokolls, das einfach gehaltene Request-Response Format, die Abwärtskompatibilität des Protokolls und noch viele weitere Punkte waren ausschlaggebend für den Erfolg des World Wide Web beziehungsweise des http. [1]

### 1.2. http-Methoden

---

Folgende http-Methoden werden verwendet wenn ein Client (Webbrowser) eine Anfrage an einen Webserver sendet. Die folgenden Methoden sind im RFC 2616 definiert worden. Jeder Webbrowser und Webserver die über die Version 1.1 kommunizieren müssen diese Methoden unterstützen.

#### 1.2.1. HTTP - GET

---

Die GET-Methode dient zur Anforderung von Daten. Diese Daten können sein ein Textdokument, ein Bild, eine Webseite oder andere Quellen. Eine Quelle wird dabei durch den Request-URL identifiziert. Generell kann man zwei Typen von GET-Methoden unterscheiden. Die „conditional GET-Methode“ und die „partial GET-Methode“. Bei der conditional GET-Methode kann die Anforderung von Daten einer bestimmten Quelle an eine oder mehrere Bedingungen geknüpft sein. Diese Bedingungen sind im Header-Feld "conditional" hinterlegt. Die vermutlich am häufigsten verwendeten Bedingungen sind „If-Modified-Since“, „If-Unmodified-Since“ und „If-Match“. Unter Verwendung dieser Bedingungen lässt sich das Datenvolumen welches übertragen werden muss minimieren. Mit diesen Bedingungen können die Anfragen so definiert werden, dass nur mehr die wirklich benötigten Daten übertragen werden müssen. Proxyserver verwenden diese Funktion, um die mehrfache Übertragung von Daten, die sich bereits im Cache befinden, zu verhindern.

Die „partial GET-Methode“ hingegen verwendet das „Range-Feld“ des http Headers. Mit diesem Feld begrenzt der Client die angeforderten Daten um nur einen Bruchteil der Daten übertragen zu müssen. Diese Technik wird verwendet wenn ein Datentransfer unterbrochen wurde, der Datentransfer wieder aufgenommen wird, um die restlichen Daten zu übertragen. [2]

#### 1.2.2. HTTP - PUT

---

Diese Methode wird verwendet um Daten auf einer bestehenden Quelle zu modifizieren beziehungsweise für die Erzeugung neuer Daten.

#### 1.2.3. HTTP - OPTIONS

---

Mit dieser Methode kann der Client Informationen über die zur Verfügung gestellten Kommunikationsoptionen abrufen. So lassen sich Beschränkungen von Quellen auf HTTP oder Proxyservern ermitteln. Diese Methode übermittelt die gewünschten Daten ohne eine Aktion oder Datentransfer einzuleiten. [2]

#### 1.2.4. HTTP – HEAD

---

Die HEAD-Methode fordert den Header von Daten einer bestimmten Quelle oder der Quelle selbst an. Der Webserver übermittelt als Antwort auf die HEAD-Anfrage die Metainformationen der angeforderten Daten. Diese Methode wird häufig verwendet um die Größe oder den Typ von Quellen beziehungsweise Objektattribute auszulesen. [2]

#### 1.2.5. HTTP – POST

---

Mit der POST-Methode werden häufig Formulareingaben an einen Webserver, Kommentare jeglicher Art und Erweiterungen von Online-Datenbanken sind mit POST möglich. Die an den Webserver übertragenen Daten befinden sich in der „Entity-Sektion“. Der von der POST-Methode übermittelte URL dient als Referenz, welche Routine auf dem Server für die Bearbeitung der Daten zuständig ist. [2]

#### 1.2.6. HTTP – DELETE

---

Mit dieser Methode werden Daten, welche durch eine URL identifiziert werden können, auf einem HTTP-Server gelöscht. Der Löschvorgang der Daten muss nicht sofort nach dem Eingang der Anfrage erfolgen. Laut RFC muss der Server lediglich die Annahme der Anforderung bestätigen. [2]

#### 1.2.7. HTTP – TRACE

---

Unter Verwendung der TRACE-Methode ist der Client in der Lage seine Anfragen zu verfolgen. Im http-Headerfeld „Via“, der Antwortnachricht, sind alle durchlaufenen Knotenpunkte mitprotokolliert. Bei Anfragen welche über mehrere Knotenpunkte (hops) laufen, wird diese Methode auch zur Fehleranalyse der Client-Server-Verbindung verwendet. [2]

#### 1.2.8. HTTP – CONNECT

---

Die CONNECT-Methode ist laut RFC 2616 für Verbindungen mit Proxyservern, welche dynamisch als Tunnel (SSL Tunnel) agieren können, reserviert. [2]

### 1.3. HTTP Header

---

Die verwendbaren Headerfelder sind für Anfrage (Request) und Antwort (Response) unterschiedlich. Die Anfrage/Antwort-Header können folgende Felder inkludieren.

#### 1.3.1. Request Header

---

Accept, Accept-Charset, Accept-Encoding, Accept-Language, Authorization, cache-Control, Connection, Cookie, Content-Length, Content-MD5, Content-Type, Date, Expect, From, Host, If-Match, If-Modified-Since, If-None-Match, If-Range, If-Unmodified-Since, Max-Forwards, Pragma, Proxy-Authorization, Range, Refer, TE, Upgrade, User-Agent, Via, Warning [2]

#### 1.3.2. Response Header

---

Accept-Ranges, Age, Allow, Cache-Control, Connection, Content-Encoding, Content-Language, Content-Length, Content-Location, Content-MD5, Content-Disposition, Content-Range, Content-Type, Date, Etag, Expires, Last-Modified, Link, Location, P3P, Pragma, Proxy-Authenticate, Refresh, Retry-After, Server, Set-Cookie, Trailer, Transfer-Encoding, Vary, Via, Warning, WWW-Authenticate [2]

Für weitere Informationen über die einzelnen http-Headerfelder verweise ich auf den RFC 2616 Kapitel 14.

### 1.4. Kommunikation mittels http

---

Bis jetzt haben wir die verwendeten Methoden und Headerfelder des http kennen gelernt. In den weiteren Kapiteln werden der Verbindungsaufbau und die verwendeten Statuscodes schematisch und anhand von Beispielen näher erklärt.

### 1.4.1. Verbindungsaufbau

Ein Client, meistens in Form eines Webbrowsers, sendet eine GET-Anfrage an den Webserver. Der Webserver verarbeitet die empfangene Anfrage direkt und liefert die gewünschten Daten mittels GET-Antwort oder der Webserver muss die Anfrage erst und sendet dann die Antwort. Diese Vorgehensweisen werden in den beiden Bildern schematisch dargestellt.



Abbildung: Direkte Verarbeitung

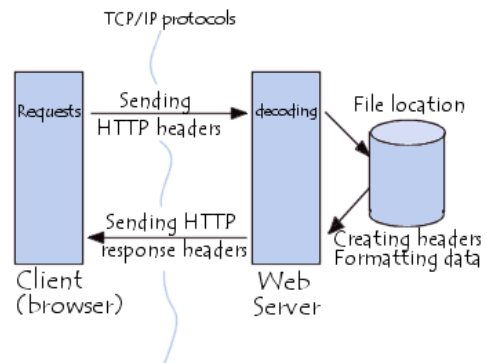


Abbildung: Indirekte Verarbeitung [http-Pic 1]

In dieser Kommunikation zwischen Client und Server werden Statuscodes verwendet, welche die Kommunikation und die Verarbeitung der Antworten erleichtern. Diese Statuscodes sind auch im RFC 2616 definiert und können in folgende Gruppen zusammengefasst werden.

### 1.4.2. Statuscodes

Die im RFC 2616 Kapitel 10 definierten Statuscodes sind in folgende Gruppen unterteilt. Die Gruppe der informellen Codes (100 und 101), der Codes für erfolgreiche Operationen (200 ... 206), der Umleitungscode (300 ... 307), der Clientfehlercodes (400 ... 417) und der Serverfehlercodes (500 ... 505).

- 1xx – Informationen
- 2xx – Erfolgreiche Operationen
- 3xx – Umleitung
- 4xx – Client-Fehler
- 5xx – Server-Fehler

Für eine genaue Auflistung und Erklärung der einzelnen Statuscodes verweise ich auf den RFC 2616 Kapitel 10. [2]

### 1.4.3. Beispiel

Mit den bekannten Methoden, Headerfeldern und Statuscodes haben wir nun genügend Informationen, um eine Kommunikation zwischen Client und Webserver im Klartext zu verstehen. Die weiteren Grafiken zeigen eine solche Kommunikation.

```
Client: GET /wiki/Spezial:Search?search=Katzen&go=Artikel HTTP/1.1
Host: de.wikipedia.org

Server: HTTP/1.0 302 Moved Temporarily
Date: Fri, 13 Jan 2006 15:12:44 GMT
Location: http://de.wikipedia.org/wiki/Katzen
...

Client: GET /wiki/Katzen HTTP/1.1
Host: de.wikipedia.org
...

Server: HTTP/1.0 200 OK
Date: Fri, 13 Jan 2006 15:12:48 GMT
Last-Modified: Tue, 10 Jan 2006 11:18:20 GMT
Content-Language: de
Content-Type: text/html; charset=utf-8

Die Katzen (Felidae) sind eine Familie aus der Ordnung der Raubtiere (Carnivora)
innerhalb der Überfamilie der Katzenartigen (Feloidea).
...
```

Abbildung: http GET-Methode [http-Pic 2-5]

## 1.5. Authentifizierung mittels http

Das http liefert auch zwei Möglichkeiten der Authentifizierung mit. Mit den beiden Methoden „basic“ und „digest access“ kann ein Server die Authentifizierung des Clients verlangen. Der Server sendet den Statuscode 401 und schreibt in das für die Authentifizierung zuständige Header-Feld den entsprechenden Wert. In der nächsten Grafik sieht man diese Vorgehensweise schematisch am Beispiel der „basic“ Authentifizierung. [7]

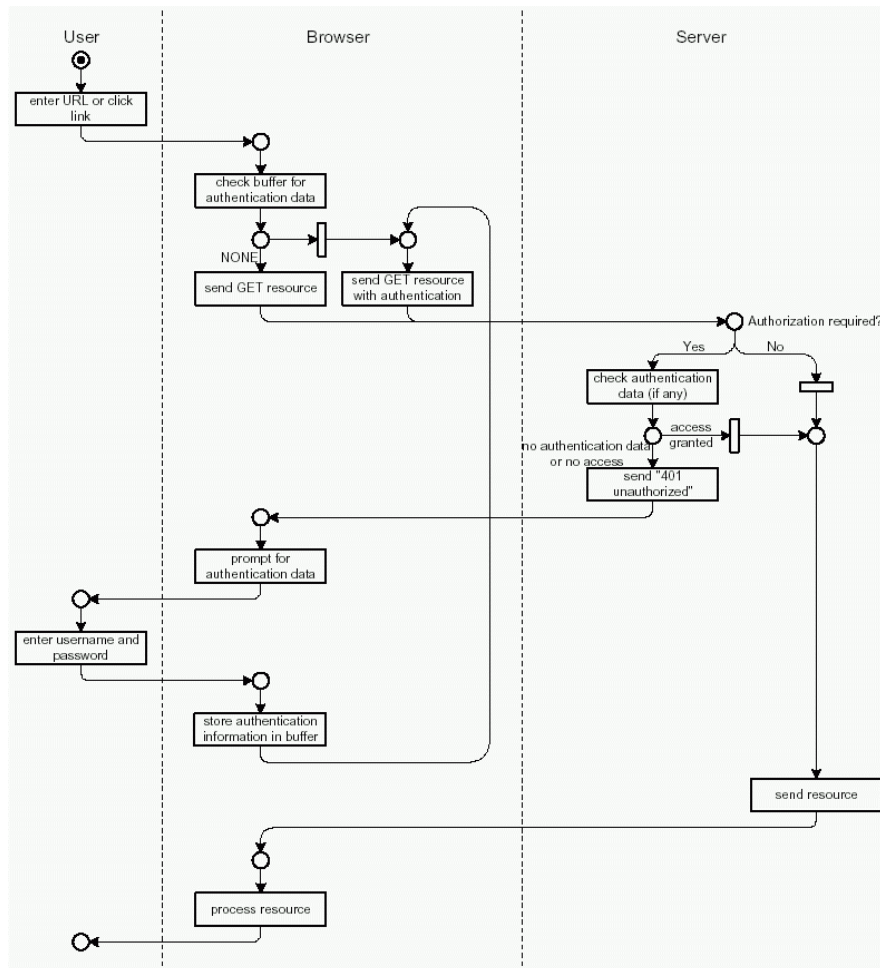


Abbildung: http-Authentifizierung [http-Pic 6]

Bei der „basic“ Authentifizierung werden allerdings Benutzername und Passwort nur codiert und ansonsten im Klartext übertragen. Die folgenden Grafiken zeigen 2 Beispiele wie einfach Benutzername und Passwort wiederhergestellt bzw. Ausgelesen werden können. In der ersten Variante sind der Benutzername Aladdin und das Passwort open sesame. Da die Daten nur Base64 codiert werden, ist es leicht möglich diese Daten zu dekodieren. Client Authentifikationsmethode mittels „basic“ und base64 Kodierung. [7]

```
GET /private/index.html HTTP/1.1
Host: localhost
Authorization: Basic QWxhZGRpbjpvYVUHNlc2FtZQ==
```

```
HTTP/1.1 200 OK
Server: HTTPd/1.0
Date: Sat, 27 Nov 2004 10:19:07 GMT
Content-Type: text/html
Content-Length: 10476
```

Abbildung: http basic Codierung [http-Pic 2-5]

Dekodierung mittels PHP.

```
echo base64_encode("Aladdin:open sesame"), PHP_EOL;
echo base64_decode("QWxhZGRpbjpvYVUHNlc2FtZQ=="), PHP_EOL;
```

Abbildung: Dekodierung der base64 codierten Benutzerdaten [http-Pic 7-9]

Im zweiten Beispiel kann man den Benutzernamen und das Passwort sofort im Klartext lesen. Das Beispiel stammt von einem Sicherheitsloch der HTC Peep-Twitter Applikation. Der folgende Teil der Kommunikation beinhaltet den Benutzernamen und das Passwort.

```

authenticity_token=c8b5abaf53f223e827d9258ddfef4285a816db5f&
oauth_token=I4FK956n1foaHjayLXXJT2IaBpsmoo0amKyPhebc&
session%5Busername_or_email%5D=USERNAME&session%5Bpassword%5D=PASSWORD

```

Abbildung: Benutzerdaten [http-Pic 7-9]

Benutzername und Passwort können in der oben gezeigten Region im Klartext gelesen werden.

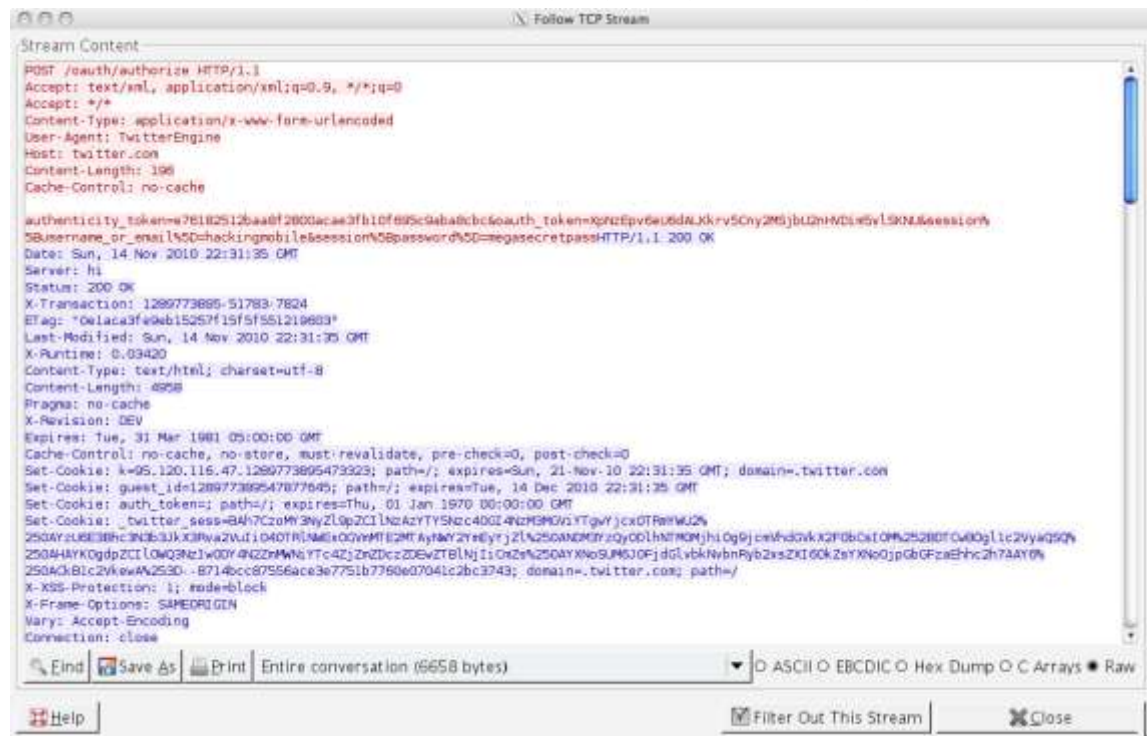


Abbildung: Ausgelesene Benutzerdaten HTC Applikation [http-Pic 7-9]

Anhand dieser beiden Beispiele kann man sehen das eine reine „basic“ Authentifizierung nicht sonderlich zur Sicherheit beiträgt. Bei der „digest access“ Authentifizierung werden Benutzername und Passwort verschlüsselt, die restliche Kommunikation wird allerdings im Klartext übertragen. Weiters ist auch die Authentifizierung mittels „digest access“ nicht sicher genug, da der client das Sicherheitsniveau bestimmt und diese Methode der Authentifizierung anfällig für „Man in the Middle“ Attacken ist. Der Client kann sich zwar beim Webserver authentifizieren, allerdings kann sich der Client nicht sicher sein ob der Webserver auch derjenige ist für den er sich ausgibt. Somit kann geschlussfolgert werden, dass eine Kommunikation mittels http unsicher ist und für eine (sichere) Kommunikation, bei der sensible Daten ausgetauscht werden, vollkommen ungeeignet ist. [7]

## 2. HTTPS (RFC 2818)

HTTPS wurde von Netscape entwickelt und in der Version 1.0 1994 vorgestellt. HTTPS wird zur Verschlüsselung der Kommunikation sowie zur Authentifikation der Kommunikationspartner verwendet. Die Verschlüsselung erfolgt mittels SSL/TLS, die Authentifikation der Kommunikationspartner erfolgt über das SSL-Handshake.

### 2.1. SSL Aufbau

HTTP verwendet SSL/TLS für die Kommunikation. SSL/TLS ist eine Zwischenschicht zwischen http und TCP/IP.

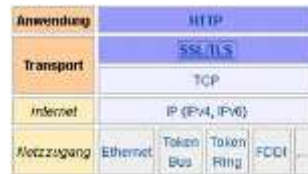


Abbildung: ISO-OSI [http-Pic 10-11]



Abbildung: SSL Aufbau [http-Pic 10-11]

Die SSL/TLS Schicht ist wie folgt aufgebaut und besteht aus den folgenden Protokollen. Für eine detaillierte Beschreibung der einzelnen Punkte verweise ich auch den RFC 2818.

### 2.2. SSL Handshake

Das SSL Handshake besteht aus 4 Phasen. In Phase 1 senden der Client und der Server ein hello. In Phase 2 identifiziert sich der Server gegenüber dem Client indem er sein X.509v3 Zertifikat sendet. Dieser Punkt kann bei „anonymen Key Agreement“ entfallen. Anschließend kann der Server optional dem Client eine Zertifikatsanfrage senden. In Phase 3 identifiziert sich der Client gegenüber dem Server und der Client versucht das vom Server empfangene Zertifikat zu verifizieren. Wenn die Verifikation fehlschlägt wird an dieser Stelle die Kommunikation beendet. In Phase 4 werden die für die verschlüsselte Kommunikation verwendeten Schlüssel abgeleitet und das Handshake beendet. Diese 4 Phasen werden schematisch in der nächsten Grafik dargestellt.

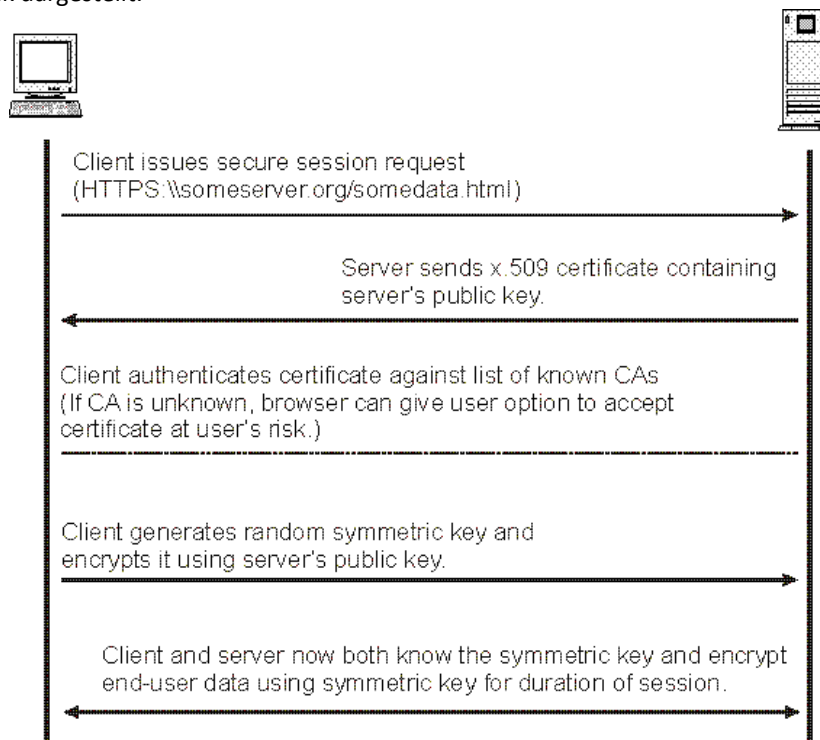


Abbildung: SSL Handshake [http-Pic 12]

public key client  
private key client

Client

public key server  
private key server

Server

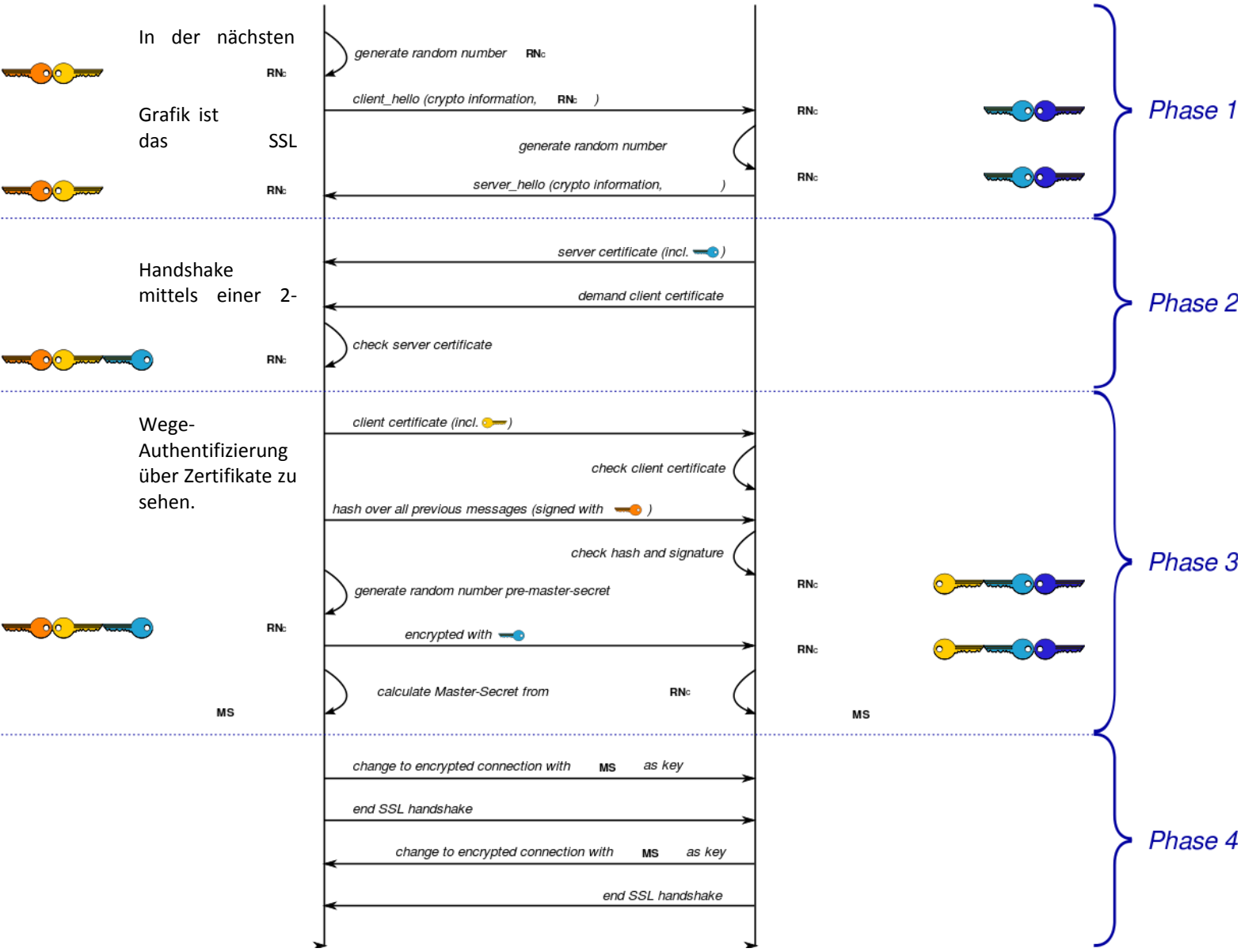


Abbildung: SSL Handshake mit Zertifikaten [http-Pic 13]

### 2.3. SSL und Zertifikate

Um die Kommunikation mittels https sicher zu machen benötigt man Zertifikate. Hier kann man für die Verwendung bei https 2 Typen unterscheiden. Die Standard X.509v3 Zertifikate und die „Extended Validation“ Zertifikate kurz EV-Zertifikate. EV-Zertifikate sind X.509 SSL Zertifikate deren Vergabe strenger kontrolliert wird. Mithilfe dieser Zertifikate sollen ungeübte User vor Phishing Attacken geschützt werden. Die Verwendung von EV-SSL Zertifikaten erkennt man den der grünen Adressleiste im Browser.



### 3. WebDAV<sup>1</sup>

---

Bei WebDAV handelt es sich um eine Erweiterung des HTTP<sup>2</sup>-Protokolls der Version 1.1. Dieses Protokoll wurde 1999 von der IETF<sup>3</sup> unter dem RFC<sup>4</sup> 2518 veröffentlicht. Es erlaubt einen kooperativen Zugriff auf Ressourcen sowohl in einem lokalen Netzwerk als auch weltweit über das Internet.

#### 3.1 Geschichte

---

Bereits in den Anfängen des öffentlich zugänglichen World Wide Web wurde festgestellt, dass das bloße HTTP-Protokoll sehr eingeschränkte Möglichkeiten in der Bearbeitung von Web Ressourcen bietet. Diese Einsicht führte zur Bildung von Forschungsteams, die sich mit der Entwicklung neuer Protokolle beschäftigten. In den folgenden Jahren veröffentlichten diese Gruppen die folgenden RFC-Dokumente, die das WebDAV-Protokoll spezifizieren und erweitern.

1999:	RFC 2518	Erste Spezifikation von WebDAV als Erweiterung von HTTP
2002:	RFC 3253	Erweiterung von WebDAV um Versionsmanagement
2003:	RFC 3648	Ordered Collections <sup>5</sup> Protocol – geordnete Datensammlungen
2004:	RFC 3744	Access Control Protocol - Zugriffsbeschränkungen
2005:	RFC 4316	Datatypes Properties - Datentypen
2006:	RFC 4331	Quotas - dient der serverseitigen Beschränkung des zur Verfügung stehenden Speicherplatzes
2006:	RFC 4437	WebDAV Ressourcenreferenzumleitung
2007:	RFC 4918	HTTP –Erweiterungen für WebDAV
2008:	RFC 5323	WebDAV-Suche

Für WebDAV wurden somit unterschiedliche Standards mit verschiedenen Schwerpunkten geschaffen, die auf einander aufbauen und die Darunterliegenden erweitern. Im Jahre 2007 wurde das Projekt als abgeschlossen angesehen und die Arbeitsgruppen schließlich aufgelöst

#### 3.2 Funktionsweise

---

WebDAV bietet durch die Anlehnung an das HTTP-Protokoll eine einfache und vergleichsweise sichere Alternative zu herkömmlichen Datenverwaltungs-Systemen wie FTP<sup>6</sup>, SAMBA<sup>7</sup>, NFS<sup>8</sup>.

Die Einrichtung gestaltet sich unkompliziert, da das Protokoll über den Port 80 ausgeführt wird, der standardmäßig für HTTP-Verbindungen geöffnet sein muss. Eine Konfiguration der Firewall ist somit nicht erforderlich. Die Sicherheit wird durch den Einsatz von Verschlüsselung der Verbindung über HTTPS erhöht. Eine solche Konfiguration ist dem Einsatz des FTP-Protokolls, welches den kompletten Informationsaustausch Klartext übermittelt, nach Möglichkeit vorzuziehen.

Als weiterer Vorteil soll nicht unerwähnt bleiben dass einem potenziellen Angreifer kein Wissen über die auf dem Server laufenden Dienste weitergegeben wird, da keine zusätzlichen Ports geöffnet werden müssen, die auf eine Freigabe und einen dezidierten Service deuten.

---

<sup>1</sup> Web-based Distributed Authoring and Versioning.

<sup>2</sup> Hypertext Transfer Protocol.

<sup>3</sup> Internet Engineering Task Force.

<sup>4</sup> Request for Comments.

<sup>5</sup> Verzeichnisse.

<sup>6</sup> File Transfer Protocol.

<sup>7</sup> Freie Software für den Zugriff auf Datei- und Druckerdienste des Server-Message-Block-Protokolls (SMB).

<sup>8</sup> Network File System.

Es muss beachtet werden, dass es Benutzern nicht möglich sein darf Skripte in Verzeichnissen auszuführen, auf die Sie selbst Zugriff über WebDAV haben. Andernfalls könnte der Server korrumpiert werden.

Das HTTP-Protokoll wird um folgende Methoden erweitert, um die nötigen Funktionalitäten, die für die Manipulation von Dateien und Verzeichnissen benötigt werden, bereit zu stellen.

COPY:	Kopieren von Dateien und Verzeichnissen, rekursive Funktion
MKCOL:	Erstellen von Verzeichnissen
MOVE:	Verschieben von Dateien und Verzeichnissen, rekursive Funktion
PROPFIND:	Abfragen von Ressourcen-Eigenschaften
PROPPATCH:	Ändern von Eigenschaften
LOCK:	Sperrern von Dateien und Verzeichnissen
UNLOCK:	Freigeben gesperrte Dateien und Verzeichnisse

Die eingeführten Methoden LOCK und UNLOCK dienen der Konfliktvermeidung und erlauben kooperatives Arbeiten. Dies ermöglicht Mehrbenutzer- und Mehrprozessorsystemen das Arbeiten auf ein und demselben Datensatz.

Das V in WebDAV steht für das im RFC 3253 spezifizierte Versionsmanagement. Es wurde von der Delta-V Arbeitsgruppe des IETF herausgegeben und ergänzt das Protokoll um weitere Methoden, die der Erfassung von Änderungen dienen.

Die Befehle eines SVN-Clients werden auf folgende interne Methoden gemappt und auf dem WebDAV-Server ausgeführt.

VERSION-CONTROL, REPORT  
CHECKOUT, CHECKIN, UNCHECKOUT  
MKWORKSPACE, UPDATE, LABEL, MERGE  
BASELINE-CONTROL, MKACTIVITY

Am Beispiel einer *svn commit*-Anweisung:

*svn commit*

„Erzeugt eine Aktivität MKACTIVITY und führt ein CHECKOUT jedes geänderten Objekts, gefolgt von einem PUT der neuen Daten, aus.“<sup>9</sup>

Das Versionsmanagement bietet nicht nur Schutz vor versehentlichem Überschreiben oder Löschen, sondern gestattet es dem Benutzer eine Datei oder ein Verzeichnis in jeden gespeicherten Zustand rückzusetzen.

### **3.3 Umsetzung / Anwendungen**

---

Das WebDAV-Protokoll wird in vielen Anwendungen umgesetzt, hier bestehen jedoch betriebssystemspezifische Unterschiede. Während unter Windows WebDAV in den Applikationen integriert wird, nutzen Programme unter Mac OS X oft das Dateisystem selbst, welches WebDAV-Spezifikationen erfüllt.

An Anwendungsmöglichkeiten mangelt es nicht, was an der breiten Fächerung von Programmen erkennbar wird, die das WebDav-Protokoll nutzen. Unentgeltliche OpenSource- Lösungen, wie cadaver, DAV Explorer, KDE Konqueror, davfs2, aber auch kommerzielle Software, wie Adobe

---

<sup>9</sup> Vgl. Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato, 2007, S.325.

Photoshop, Microsoft Office, Novell NetDrive, Microsoft Web Folders oder Mac OS X unterstützen WebDAV.

#### 4. CalDAV

Das CalDAV ist ein offenes Client-/Server-Protokoll und dient der Kalenderverwaltung und der Terminplanung. Es basiert auf der Grundspezifikation von WebDAV (RFC2518) und dessen Erweiterung WebDAV ACL (veröffentlicht unter RFC 3744). CalDAV verwendet für die Übertragung und Administration von Terminen und Kalendern das textbasierte iCalendar-Format. Das Protokoll ermöglicht den Austausch von Kalenderdaten zwischen unterschiedlichen Betriebssystemen, Terminverwaltungsprogrammen und über Organisationsgrenzen hinweg.

##### 4.1 Geschichte

Die Entwicklung von CalDAV wurde 2003 durch einen Entwurf von Lisa Dusseault, der an die IETF übermittelt wurde, angeregt. Das Protokoll wurde 2007 von der IETF unter der Kennnummer RFC 4791 veröffentlicht.

##### 4.2 Funktionsweise

Der direkte Austausch von Terminen erfolgt über das Transport-Independent Interoperability Protocol (iTIP) RFC 2446. Im Gegensatz dazu wird iMIP (RFC 2447) zur Übertragung über das Mail-Protokoll eingesetzt.

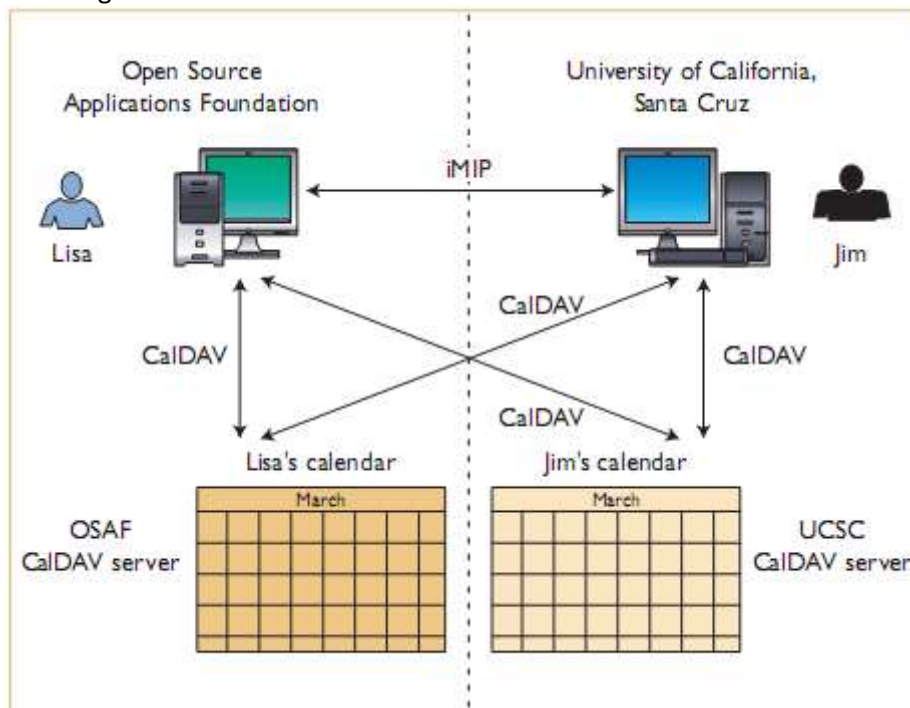


Abbildung: CalDAV Terminaustausch über iMIP und iTIP<sup>10</sup>

<sup>10</sup> Quelle: Lisa Dusseault, Jim Whitehead, 2005, S.83.

Das verwendete iCalendar-Format kann jeweils einen der folgenden Termin-Typen beinhalten; VEVENT, VTODO, VJOURNAL, VFREEBUSY.

Werden unterschiedliche Typen benötigt, so müssen diese über Collections, die eine Sammlung von einzelnen iCalendar-Ressourcen darstellen, gebündelt werden.

Möchte eine Person einen Termin mit einem Benutzer vereinbaren, so sendet er eine Anfrage an den CalDAV-Server des Benutzers die in der sogenannten *Inbox* abgelegt wird und auf Bestätigung wartet. Falls der Benutzer nicht verfügbar ist beantwortet der Server dies mit einer Fehlermeldung und sendet die *Busy-Time* damit der Absender seine Terminanfrage neu formulieren kann. Termine, die sich in der *Inbox* befinden gelten als vorläufige Termine und werden bei der Planung und Beantwortung neuer Anfragen berücksichtigt.

Des Weiteren können Updates über IF-Klauseln spezifiziert werden, hier kann der Benutzer Bedingungen angeben die erfüllt sein müssen damit ein Update durchgeführt wird und was im Falle einer Verletzung der Bedingung zu geschehen hat.

CalDAV implementiert keinerlei Sicherheitsmechanismen und überlässt diese dem Entwickler und den darunterliegenden Schichten.

Zu beachten sind an dieser Stelle unter anderem die Übertragung in Klartext, Zugriffsrichtlinien und Prozeduraufrufe als Alarmaktionen.

Abhilfe in der Klartextproblematik kann die Übertragung über eine sichere Verbindung mittels VPN oder HTTPS bringen. Zugriffsrichtlinien werden durch hierarchische Benutzerkonten in WebDAV ermöglicht, sprich wer darf welche Einträge lesen, ändern und erstellen. Prozeduraufrufe als Alarmaktionen dürfen nur von Administratoren gesetzt werden, um Denial-Of-Service –Attacken zu verhindern.

### 4.3 Umsetzung / Anwendungen

---

Das CalDav-Protokoll kommt beinahe unscheinbar in viele Applikationen des täglichen Gebrauchs eines jeden Computernutzers zum Einsatz. Applikationen die bereits heute CalDAV als Clienten umsetzen sind unter anderem Evolution, Sunbird, Microsoft Outlook über ZideOne Plugin, CalDAV-Sync für Android, iCal, iPhone.

Zu CalDAV-Servern zählen zum Beispiel der Yahoo Kalender, Sun Java Calendar Server, DavMail, Google Calendar, eGroupWare, Apple Darwin Calendar Server und viele andere.

## 5. Literaturverzeichnis

---

1. [http://www.tecchannel.de/netzwerk/management/401210/hypertext\\_transfer\\_protocol/](http://www.tecchannel.de/netzwerk/management/401210/hypertext_transfer_protocol/)
2. <http://www.ietf.org/rfc/rfc2616.txt>
3. [http-Pic 13]; <https://developer.connectopensource.org/display/CONNECTWIKI/SSL+Handshake>
4. <http://svnbook.red-bean.com/de/1.5/svn.webdav.clients.html#svn.webdav.clients.tbl-1>
5. [http-Pic 12];  
[http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=%2Fcom.ibm.itame2.doc\\_5.1%2Fss7aumst18.htm](http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=%2Fcom.ibm.itame2.doc_5.1%2Fss7aumst18.htm)
6. <http://greenbytes.de/tech/webdav/common-index.html>
7. [http://en.wikipedia.org/wiki/Digest\\_Access\\_Authentication](http://en.wikipedia.org/wiki/Digest_Access_Authentication)
8. [http-Pic1];  
<http://www.google.at/imgres?q=geschichte+des+http+protokolls&um=1&hl=de&sa=N&biw=1680&bih=935&tbm=isch&tbnid=3P3FbjWkVb94iM:&imgrefurl=http://de.kioskea.net/contents/internet/http.php3&docid=Mii2OYznXJdVYM&imgurl=http://static.commentcamarche.net/de.kioskea.net/pictures/internet-images-comm.gif&w=377&h=270&ei=PtLUTq-eL8fm-gagppSnDw&zoom=1&iact=hc&vpx=669&vpy=486&dur=3350&hovh=190&hovw=265&tx=118&ty=101&sig=117333153359690505351&page=1&tbnh=128&tbnw=179&start=0&ndsp=43&ved=1t:429,r:29,s:0>
9. [http-Pic 2-5]; [http://de.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](http://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol)
10. <http://www.ietf.org/rfc/rfc2818.txt>
11. <http://www.ietf.org/rfc/rfc2518.txt>
12. <http://www.ietf.org/rfc/rfc4791.txt>
13. [http-Pic 7-9]; <http://blog.taddong.com/2011/02/vulnerability-in-htc-peep-twitter.html>
14. [http-Pic 6]; [http://www.fmc-modeling.org/category/projects/apache/amp/2\\_4Access\\_Control.html](http://www.fmc-modeling.org/category/projects/apache/amp/2_4Access_Control.html)
15. [http-Pic 10-11]; [http://de.wikipedia.org/wiki/Transport\\_Layer\\_Security](http://de.wikipedia.org/wiki/Transport_Layer_Security)
16. **Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato** Versionskontrolle mit Subversion, O'Reilly Verlag, Auflage: 1. A., April 2005; überarbeiteter Nachdruck 2007. (ISBN-13: 978-3897214606).
17. **Lisa Dusseault, Jim Whitehead**, IEEE INTERNET COMPUTING 1089-7801/05/\$20.00 © 2005 IEEE Published by the IEEE Computer Society MARCH • APRIL 2005