# Nutzung von selbstsignierten Client-Zertifikaten zur Authentifikation bei SSL/TLS

Tibor Jager
tibor.jager@rub.de

Heiko Jäkel
heiko.jaekel@udo.edu

Jörg Schwenk
joerg.schwenk@rub.de

**Abstract:** Für die sichere browserbasierte Kommunikation im Internet wird heutzutage häufig das SSL-Protokoll benutzt. In der Regel weist sich der Server dabei mit einem von einer vertrauenswürdigen Partei (CA) signierten Zertifikat gegenüber dem Benutzer aus. Der Benutzer authentifiziert sich über einen Login mit Benutzername und Passwort. Diese Form der Authentifizierung ist allerdings anfällig für Attacken wie Phishing, Pharming und Identitätsdiebstahl, da sich jeder Angreifer, der in den Besitz des Passworts kommt, erfolgreich damit bei dem Server authentifizieren kann. Ausserdem zeigt die Erfahrung, dass viele Benutzer nicht darauf achten, ob ihr Passwort stets über eine sichere Verbindung übertragen wird. Um diesen Risikofaktor zu eliminieren, kann sich ein Benutzer bei SSL auch mit einem Zertifikat gegenüber dem Server authentifizieren. Dies wird allerdings in der Praxis selten genutzt, da kaum ein Webservice diese Möglichkeit anbietet und nur die wenigsten Benutzer über ein Zertifikat von einer kommerziellen Certification Authority (CA) verfügen. Wir präsentieren in dieser Arbeit einen Ansatz zur Verwendung von *selbstsignierten* Client-Zertifikaten, welcher im Wesentlichen darauf basiert dass der Benutzer sich selbst ein Zertifikat ausstellt und sich damit beim Server authentifiziert. Beispielhaft wurde hierfür ein Plugin für den Firefox-Browser [Moz07] implementiert und mit dem Apache-Webserver [Apa07] getestet.

**Keywords**: Zertifikate, Authentifikation, SSL

## 1 Einleitung

Wenn sich ein Benutzer im Internet gegenüber einem Webserver authentifizieren möchte, so geschieht dies meist über eine Benutzername-Passwort-Abfrage. Dies hat allerdings einige Nachteile. Viele Benutzer neigen dazu, sehr einfache Passworte zu verwenden, die einfach zu erraten und anfällig gegen Dictionary-Attacken sind [MT79]. Wenn das Passwort des Benutzers einmal einem Angreifer bekannt ist, kann dieser sich immer wieder erfolgreich gegenüber dem Server authentifizieren.

Für sicherheitsrelevante Anwendungen, wie Onlinebanking, Internetauktionen und generell für Anwendungen bei denen der Benutzer ein Passwort eingeben muss, sollte stets nur ein vertrauenswürdiges System genutzt werden. Damit ist beispielsweise sicheres Onlinebanking auf einem fremden PC-System (z.B. Internetcafe) nicht möglich. Da daher viele Benutzer immer denselben PC (Zuhause, Arbeitsplatz) für sicherheitsrelevante Anwendungen benutzen, bietet es sich an, den Anwender mit dem genutzten PC bzw. Browser zu „verbinden“. Dies kann über ein Zertifikat geschehen, welches im Browser gespeichert

und für die Authentifikation gegenüber dem Server verwendet wird.

Die Verwendung identischer Passworte für verschiedene Webdienste sollte stets vermieden werden, da ansonsten ein Angreifer, der in Besitz des Passwortes von einem Webdienst kommt, auch Zugriff auf alle anderen Dienste erlangt. Bei der Verwendung mehrerer passwortgeschützter Webdienste sollte also stets ein individuelles Passwort für jeden Dienst verwendet werden. Das bedeutet, dass sich der Benutzer zahlreiche verschiedene Passworte merken muss.

Beim Einsatz von selbstsignierten Client-Zertifikaten genügt es, den persönlichen Schlüsselbund des Benutzers mit einem Master-Passwort abzusichern. Für jeden Webdienst kann (zur Gewährleistung von Nicht-Verknüpfbarkeit) ein eigenes Client-Zertifikat erstellt werden. Jedoch auch die Verwendung des gleichen Client-Zertifikates für verschiedene Webdienste ist möglich. Für den Benutzer besteht der Vorteil darin, dass er sich nicht zahlreiche verschiedene Passworte merken und eingeben muss, sondern nur noch ein Master-Passwort welches den persönlichen Schlüsselbund des Benutzers absichert. Zusätzlich wird die Sicherheit der Verbindung erhöht, denn ein Angreifer müßte schon in den Besitz des privaten Schlüssels kommen, um sich gegenüber dem Server zu authentifizieren.

Die Idee, selbstsignierte Zertifikate im Browser ohne CA zu erzeugen lieferte die Arbeit zur Serveridentifikation mit einem visuellen Artefakt (VA), die in [GSW06] vorgestellt wurde. Dabei identifiziert sich der Client gegenüber dem Server mit einem Zertifikat. Der Benutzer speichert vorab beim Server ein Bild ab, welches ihm nach der Übertragung des Zertifikats vom Server gezeigt wird. Wenn der Benutzer dieses VA erkennt, weiß er, dass er mit dem richtigen Server verbunden ist. Dann kann er sich wie gewohnt mit seinem Benutzernamen und Passwort einloggen. In diesem Zusammenhang stellte sich die Frage, wie der Benutzer an ein Zertifikat kommen soll.

In dieser Arbeit stellen wir einen Ansatz vor, um den unsicheren Faktor „Benutzer“ aus dem Login-Prozess herauszunehmen. Dazu wurde ein Plugin für den Firefox Browser implementiert, mit dem ein unerfahrener Benutzer sich selbst ein Zertifikat ausstellen kann. Ein wichtiger Faktor ist dabei auch die einfache Bedienung. Untersuchungen zeigen [WT99], dass beim Umgang mit Sicherheitssoftware viele Fehler gemacht werden können und dass nicht die Software, sondern der, der diese bedient, der größte Risikofaktor ist. Grundlegende Zielsetzung bei der Entwicklung des Ansatzes ist die einfache Integration in die vorhandene Infrastruktur. Das bedeutet insbesondere, daß keine Modifikationen am Handshake-Protokoll von SSLv3/TLS [1] notwendig sind. Ausserdem verwenden wir den Apache-Webbrowser mit dem SSL-Modul mod_ssl für die Referenzimplementierung.

Der von uns vorgestellte Ansatz erreicht sowohl einen Sicherheitsgewinn, da die Komponente “Benutzer” aus dem Login-Prozess ausgeschlossen wird, als auch eine Verbesserung des Benutzerkomforts, da sich der Anwender nur noch ein Master-Passwort für den Schlüsselbund merken muss. Trotzdem können verschiedene Schlüssel für verschiedene Webdienste verwendet werden (Um dies mit herkömmlicher Benutzername-Passwort Authentifikation zu erreichen müsste sich der Anwender ein eigenes Passwort für jeden genutzten Webdienst merken). Das ist bemerkenswert, da im Regelfall Sicherheitsgewinne

[1] SSLv3 und TLS verwenden beide das gleiche Handshake-Protokoll. Wir verwenden daher nachfolgend der Einfachheit halber den Begriff *SSL* zusammenfassend für SSLv3 und TLS.

nur auf Kosten der Benutzbarkeit realisiert werden können.

## 2 Client-Zertifikate in SSL/TLS

Die Nutzung von Client-Zertifikaten ist bereits als optionale Variante im SSL-Handshake vorgesehen. Nach der `server_hello`-Nachricht sendet der Server (optional) sein von einer vertrauenswürdigen CA signiertes Zertifikat, und hat dabei auch die Möglichkeit ein Zertifikat vom Client anzufordern (vgl. Abb. 1).

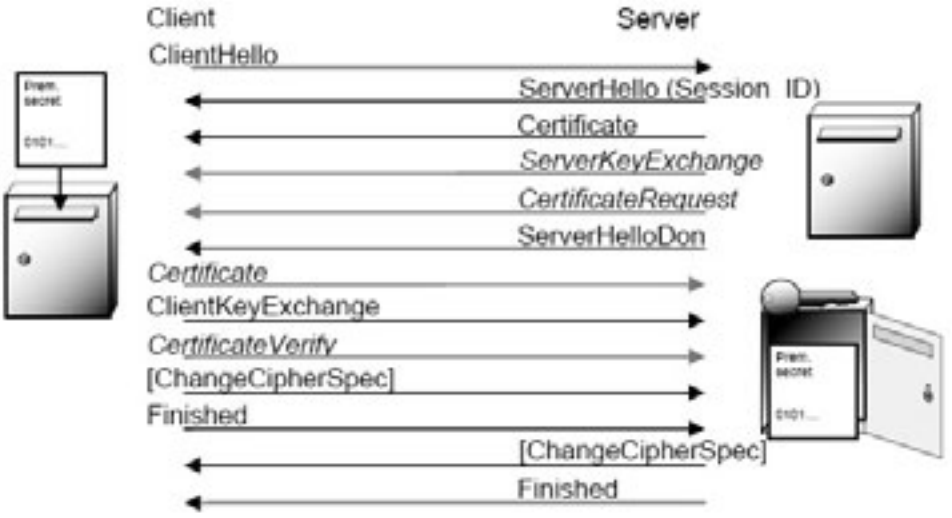


Abbildung 1: Das SSL Handshake-Protokoll, Quelle: [Sch05]

Im Apache-Modul `mod_ssl` [Mod07] sind vier Varianten der Client-Authentifikation vorgesehen (vgl. Abb. 2). Wenn die Direktive `SSLVerifyClient` auf `none` gesetzt ist, so ist keine Clientauthentikation vorgesehen. Durch den Parameter `optional` wird konfiguriert, dass der Client *optional* ein Zertifikat vorlegen kann. Das vorgelegte Zertifikat wird mittels einer internen Liste gültiger Root-Zertifikate validiert. Die Einstellung `require` bewirkt, dass ein gültiges Zertifikat zwingend verlangt wird. Auch in diesem Fall wird das Zertifikat über die interne Root-CA-Liste validiert. Die vierte mögliche Variante ist `optional_no_ca`. Hier kann ein beliebiges Zertifikat vorgelegt werden, welches nicht validiert wird. Wie auch die `mod_ssl` Dokumentation [Mod07] anmerkt, widerspricht die letztgenannte Variante natürlich (zunächst) dem Gedanken der Authentifizierung, denn es kann ein beliebiges Zertifikat vorgelegt werden.

Bei den Optionen `optional` und `require` validiert der Server das erhaltene Client-Zertifikat mithilfe einer interne Liste von gültigen Root-CA-Zertifikaten. Diese SSL-Varianten sind also zur Client-Authentikation mit Zertifikaten, die von einer gemeinsamen

| | |
|---|---|
| `none` | Es ist kein Client-Zertifikat vorgesehen |
| `optional` | Der Client *kann* optional ein gültiges Zertifikat vorlegen |
| `require` | Der Client *muss* ein gültiges Zertifikat vorlegen |
| `optional_no_ca` | Der Client *kann* ein Zertifikat vorlegen, es wird jedoch nicht prüft ob das Zertifikat von einer gültigen CA stammt |

Abbildung 2: Betriebsarten der `mod_ssl`-Direktive `SSLVerifyClient`

*vertrauenswürdigen CA* ausgestellt wurden, geeignet .

## 3 Selbstsignierte Client-Zertifikate in SSL/TLS

In diesem Abschnitt beschreiben wir die Verwendung von *selbstsignierten* Client-Zertifikaten zur Browser-Authentikation in SSL. Ein natürlicher Implementierungsansatz ist, die in SSL zur Clientauthentikation vorgesehenen *CA-signierten* Zertifikate einfach durch *selbstsignierte* Zertifikate zu ersetzen.

Dazu muss die Validierung der selbstsignierten Zertifikate bereits im SSL-Handshake realisiert werden. Damit der Server ein empfangenes Client-Zertifikat als authentisch anerkennt, muss normalerweise das zugehörige root-Zertifikat dem Server bekannt sein. Da in diesem Fall das Client-Zertifikat auch root-Zertifikat ist, muss dieses vom Server in die Liste seiner root-Zertifikate aufgenommen werden.

Dieser natürliche Ansatz stösst jedoch schnell an seine Grenzen, insbesondere weil für jeden neuen Benutzer die Liste der vom Server als vertrauenswürdig anerkannten Root-CAs um einen Eintrag erweitert werden muss. Dies ist in der Praxis nicht ohne Weiteres zu realisieren: Wenn der Webserver so konfiguriert ist, dass ein Client-Zertifikat zwingend verlangt wird (d.h. beispielweise beim Apache Modul `mod_ssl` die Direktive `SSLVerifyClient=require` gesetzt ist), dann schickt der Server zunächst eine Liste von vertrauenswürdigens CAs an den Benutzer. Da für jedes Client-Zertifikat (genauer: Für jede neue Benutzer-CA) eine neues Root-Zertifikat installiert wird, wird diese Liste mit jedem Benutzer länger. Die Größe der entsprechenden Nachricht ist aber laut SSL-Spezifikation (siehe [DA99]) auf 65535 Bytes limitiert. Ab einer gewissen Anzahl von installierten Benutzer-CAs ist es also nicht mehr möglich neue Benutzer aufzunehmen.

Während die Ursache des letztgenannte Problems im SSL Handshake-Protokoll liegt, ergibt sich speziell beim Einsatz des Apache Webservers eine weitere Einschränkung: Der Apache-Webserver mit `mod_ssl` [Mod07] lädt die Liste der vertrauenswürdigen Zertifikate erst bei einem Neustart neu. Es reicht zwar ein *graceful*-Restart, bei dem aktive Verbindungen erhalten bleiben, aber trotzdem ist es in der Praxis inakzeptabel, dass eine Aktion eines Benutzers einen Neustart des Servers anstößt. Die *Authentizität* des Zertifi-

kats kann also nicht im SSL-Handshake validiert werden.

Um beispielsweise bei Verwendung des Apache-Webservers die zuvor genannten Probleme zu umgehen muss also zunächst die `mod_ssl`-Direktive `SSLVerifyClient` auf `optional_no_ca` gesetzt werden. Mit dieser Option kann der Benutzer im SSL-Handshake ein Zertifikat an den Server senden, ohne dass dieser zuvor eine Liste der vertrauenswürdigen CAs an den Client schicken muss. Es wird nur überprüft, ob der Client auch im Besitz des privaten Schlüssels ist, jedoch nicht die Authentizität des Zertifikats validiert.

Da weiterhin die *Authentizität* des Zertifikats nicht im SSL-Handshake validiert werden kann, muss dies unmittelbar danach geschehen. Wir erweitern daher den Server um eine Datenbank, in der Client-Zertifikate gültigen Benutzeridentitäten zugeordnet werden. Nach dem SSL-Handshake vergleicht der Server das erhaltene Zertifikat mit den in der Datenbank gespeicherten. Falls das erhaltene Zertifikat einem in der Datenbank gespeicherten entspricht, kann der Server dann annehmen, dass es sich um einen legitimen Benutzer handelt, da bereits im SSL-Handshake sichergestellt wurde, dass der Client im Besitz des zugehörigen privaten Schlüssels ist.

Bislang setzen wir voraus, dass die Client-Zertifikate von gültigen Benutzern bereits in einer Datenbank auf dem Server gespeichert sind. Die Zertifikate müssen also zuvor über einen vertrauenswürdigen Kanal an den Server übertragen worden sein. Eine sehr einfache Variante ist beispielsweise, dass der Benutzer bei seiner Registrierung auf dem Server das Client-Zertifikat mit übermittelt. Dies entspricht dem in weniger sicherheitskritischen Anwendungen üblichen Vorgehen, bei dem ein Benutzer bei der Registrierung selbst ein Passwort wählt über das er später authentifiziert wird. Für Anwendungen mit höheren Anforderungen lässt sich diese Vorgehensweise leicht erweitern, beispielsweise indem der Benutzer den sha1-Fingerprint des Zertifikats zusammen mit einem Legitimationsnachweis (z.B. Personalausweis) über einen sicheren Kanal an den Server übermittelt. Dazu bietet sich für praktische Anwendungen beispielsweise das PostIdent-Verfahren [Deu07] an.

Das zuvor beschriebene Verfahren stellt eine einfache Ergänzung zu den SSLv3/TLS-Standards [DA99] dar. Insbesondere lässt sich die Erweiterung ohne Modifikationen am SSL-Protokoll umsetzen. Es ist lediglich ein weiterer Schritt zur Verifikation des Zertifikats nötig, bei dem das erhaltene Zertifikat mit den gespeicherten gültigen Client-Zertifikaten der Benutzer abgeglichen wird. Diese Schritt ersetzt den zuvor notwendigen Schritt der Passwortverifikation.

Das beschriebene Verfahren erhöht zum einen die Sicherheit, da es nicht mehr notwendig ist darauf zu vertrauen, dass der Benutzer selbst sichere Passworte wählt. Gleichzeitig erhöht das Verfahren die Benutzerfreundlichkeit, da der Benutzer sich anstatt zahlreicher verschiedener Passworte für verschiedene Webdienste nur noch das Master-Passwort seines persönlichen Schlüsselbundes merken muss.

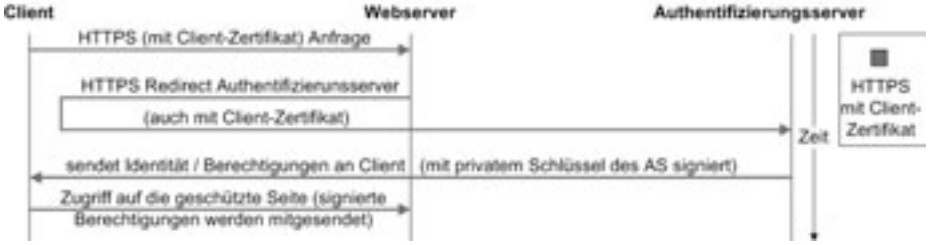


Abbildung 3: Automatischer Sign-On mit einem Client-Zertifikat

## 4 Automatischer Sign-On mit Client Zertifikaten

Nachfolgend beschreiben wir den Einsatz von selbstsignierten Client-Zertifikaten zur Realisierung eines automatisierten Sign-On, welcher sich gut zur Implementierung von Single Sign-On eignet. Zu diesem Zweck betrachten wir ein Modell mit drei Parteien: Es gibt den Benutzer, der im Besitz eines selbstsignierten Client-Zertifikat ist, einen Webserver, dessen Dienst der Benutzer in Anspruch nehmen möchte, und einen Authentifizierungsserver (AS).

Bevor der Anwender den Authentifizierungsdienst nutzen kann, muss er zunächst ein selbstsigniertes Zertifikat erzeugen (zum Beispiel mit einem Browser-Plugin, vgl. Abschnitt 5.1), und dieses über einen vertauenswürdigen Weg an den AS übermitteln (vgl. Abschnitt 3).

Wenn der Benutzer nun einen Webdienst in Anspruch nehmen will, der den Authentifizierungsdienst als vertrauenswürdig anerkennt, verbindet er sich mit diesem und übermittelt dabei sein selbstsigniertes Zertifikat. Dies kann z.B. durch einen zusätzlichen Button auf der Webseite des Dienstes realisiert werden, der sich neben dem normalen Login-Feld befindet, in das man sonst den Benutzernamen und das Passwort eingibt.

Über einen HTTPS-Redirect wird der Anwender nun auf den AS umgeleitet, welcher ebenfalls das Client-Zertifikat überprüft und bei Erfolg die Identität bzw. die Rechte dieses Benutzers zurückschickt. An dieser Stelle können z.B. kurzlebige Attribut-Zertifikate oder Cookies verwendet werden, die mit dem privaten Schlüssel des AS signiert sind. Der Webserver kann auch vorher mit dem AS einen symmetrischen Schlüssel über einen externen Kanal tauschen und diesen für den Attributaustausch verwenden. Die empfangenen Attribute kann der Anwender nun an den Webserver schicken und so einen vertraulichen Dienst von diesem in Anspruch nehmen. Wichtig ist, dass die Attribute zusammen mit dem öffentlichen Schlüssel des Anwenders verschickt werden und diese beiden Dinge nicht voneinander trennbar sind. Der Ablauf dieses Verfahrens wird noch einmal in Abbildung 3 veranschaulicht.

Das beschriebene Verfahren lässt sich leicht zur Implementierung von Single Sign-On erweitern, da sich der Benutzer jeweils nur gegenüber dem AS authentifiziert. Dies setzt jedoch voraus, dass alle genutzten Services den AS als vetrauenswürdige Instanz anerkennen.


Abbildung 4: Das Plugin für Firefox zur Zertifikatsgenerierung

## 5 Benutzerfreundlichkeit und Anwendbarkeit des Verfahrens

Um die Benutzerfreundlickeit des beschriebenen Sign-On Verfahrens zu belegen, wurde ein Plugin für den Firefox-Webbrowser [Moz07] entwickelt, mit dem Client-Zertifikate auf benutzerfreundliche Art und Weise generiert werden können.

Das implementierte Plugin ist so einfach wie möglich gehalten, so dass ein durchschnittlicher Internet-Benutzer, auch wenn er keine Erfahrung mit Zertifikaten und Sicherheitssoftware hat, sich selbst ein Zertifikat ausstellen kann. Dabei wird bewußt auf Einstellungsmöglichkeiten verzichtet, die den Benutzer verwirren könnten. Im folgenden Abschnitt wird das Plugin kurz beschrieben.

### 5.1 Das Plugin für Firefox

Das Plugin nutzt eine Bibliothek des bereits verfügbaren Plugins Key Manager um auf die Network Security Services (NSS) von Firefox zuzugreifen. Zur Erzeugung des Zertifikats muss der Anwender lediglich einen Namen für das Zertifikat vergeben und auf Knopfdruck wird das Zertifikat erzeugt und in den Browser importiert. Der eingegebene Name wird für den Common Name (CN) des Zertifikats benutzt. Alle weiteren Einträge des Distinguished Names (DN) werden mit zufälligen Zeichenketten gefüllt. Abbildung 4 zeigt das Plugin. Die Gültigkeit des Zertifikats wird automatisch auf 3 Monate gesetzt. Der DN ist in diesem Fall zu vernachlässigen, da die einzig wichtige und verwendbare Information das asymmetrische Schlüsselpaar darstellt. Der Benutzer sollte allerdings einen einprägsamen Namen für das Zertifikat vergeben, da er bei der Zertifikats-Anfrage des Servers dieses aus der Liste seiner persönlichen Zertifikate auswählen muss, wie im nächsten Abschnitt beschrieben wird.

### 5.2 Zertifikats-Anfrage des Servers

Da im praxistauglichen Ansatz davon ausgegangen wird, dass des Server im SSL-Handshake keine Liste seiner vertrauenswürdigen CAs an den Client sendet, kann der Benutzer also jedes beliebige Zertifikat an den Server senden. Das entsprechende Dialogfenster zeigt Abbildung 5. Wenn der Benutzer nur ein Zertifikat in seinem Browser gespeichert

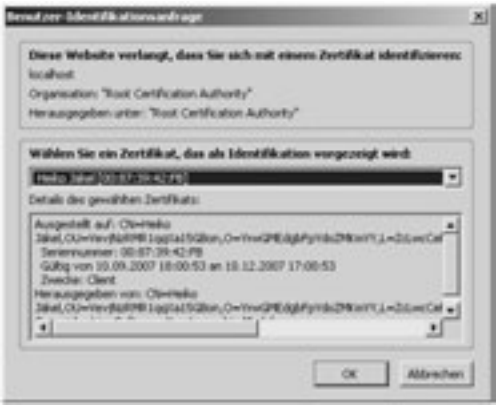


Abbildung 5: Zertifikats-Anfrage des Servers

hat, kann er einstellen, dass der Browser dieses bei einer Zertifikats-Anfrage automatisch sendet. Falls er allerdings mehrere selbstsignierte Zertifikate erzeugt hat, muss das zum jeweiligen Service gehörende ausgewählt werden. Dies lässt sich leicht realisieren, indem beispielsweise mit dem Zertifikat die URL des zugehörigen Dienstes gespeichert wird.

## 6 Gültigkeitsdauer, Sperrung und Erneuerung von selbstsignierten Zertifikaten

Ein wichtiger Aspekt im Zusammenhang mit Public-Key-Krytographie ist die Gültigkeitsdauer bzw. die Sperrung von Zertifikaten. Der X.509-Standard sieht für die Validierung und Sperrung *Certificat Revocation Lists* (CRL) oder andere Methoden wie das *Online Certificate Status Protocol* (OCSP) vor.

Bei selbstsignierten Zertifikaten ist, wie schon erwähnt, die Nutzung von CRLs nicht möglich, denn dafür müsste eine CA vorhanden sein, die diese CRL herausgibt. Da in diesem Fall der Inhaber des Zertifikats selber die CA darstellt, ist die Nutzung einer CRL nutzlos. Falls ein Zertifikat zurückgezogen werden soll, z.B. weil der private Schlüssel bekannt geworden ist, so muss der Eigentümer selbst veranlassen, dass das Zertifikat dort, wo es eingetragen ist, zurückgezogen bzw. gelöscht wird. Dies ist sicherlich ein großes Problem von selbstsignierten Zertifikaten, denn wenn das Zertifikat bei mehreren Stellen registriert ist, muss jede Stelle einzeln über die Sperrung informiert werden. Die Nutzung von CRLs in gewöhnlichen PKI-Strukturen ist allerdings auch nicht 100% sicher, da CRLs nur in regelmäßigen Abständen erneuert werden und es dadurch eine Weile dauern kann, bis die Sperrung eines Zertifikats öffentlich bekannt ist.

Ein weiterer wichtiger Aspekt ist der Zeitraum, für den das Zertifikat gültig sein soll. Auf der einen Seite steigt mit der Länge der Gültigkeit und der Häufigkeit der Benutzung die Gefahr, dass der private Schlüssel von einem Angreifer ausgespäht werden kann und auf

der anderen Seite kann eine zu kurze Gültigkeit Ärger beim Benutzer hervorrufen, denn dieser möchte sein Zertifikat eine Zeit nutzten, ohne ein neues zu erzeugen und registrieren zu lassen. Trotzdem sollte bei den in dieser Arbeit verwendeten Zertifikaten keine zu lange Gültigkeitsdauer vergeben werden.

Dies führt auch direkt zu dem dritten Aspekt, der Erneuerung, das heißt der Verlängerung der Laufzeit eines Zertifikats. Da es keine CA gibt, die das Zertifikat erneuern kann, muss der Inhaber das Zertifikat selbst erneuern. Es gibt also keine Kontrolle über die Gültigkeit, da der Inhaber sein Zertifikat beliebig oft erneuern kann. Man kann dem Zertifikat also theoretisch schon bei der Erstellung des Zertifikats eine sehr lange Gültigkeit geben. Anstatt ein Zertifikat zu erneuern, sollte es deshalb die Möglichkeit geben, ein neues selbstsigniertes Zertifikat registrieren zu lassen. Dazu muss der Benutzer eine neues Zertifikat mit seinem Browser erzeugen und sich seinem noch gültigen Zertifikat bei dem entsprechenden Dienst einloggen. Dort sollte dem Anwender die Möglichkeit gegeben werden, ein neues Zertifikat „anzumelden“. Es besteht auch die Möglichkeit, dass der entsprechende Dienst automatisch ein neues Zertifikat vom Benutzer anfordert, wenn die Gültigkeit des momentan aktuellen sich dem Ende zuneigt.

## 7 Fazit

In dieser Arbeit wurde eine Möglichkeit vorgestellt, wie ein unerfahrener Benutzer sich selbst mit seinem Browser (in diesem Fall Firefox) ein Zertifikat ausstellen und dieses zur Authentifikation genutzt werden kann.

Wenn ein solches Zertifikat einmal bei einem Webdienst registriert ist, bietet es dem Benutzer mehr Komfort und Sicherheit als der übliche Login mit einem Passwort. Erstens besteht dabei nicht die Gefahr, dass ein Angreifer, der in den Besitz des Passwortes gekommen ist, sich fälschlich als der rechtmäßige Benutzer einloggen kann und zweitens wird der Benutzer nicht dazu verleitet, ein einfach zu erratendes Passwort zu benutzen. Von Vorteil ist auch, dass es bei dem vorgestellten Ansatz keine Probleme mit PKIs und Vertrauensbeziehungen auftreten, weil keine Zertifikatsketten oder Ähnliches verwendet wird.

## Literatur

[Apa07] Apache Software Foundation. The Apache HTTP Server Project. http://httpd.apache.org/, 2007.

[DA99] T. Dierks und C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), Januar 1999. Obsoleted by RFC 4346, updated by RFC 3546.

[Deu07] Deutsche Post AG. PostIdent Verfahren. http://www.deutschepost.de/, 2007.

[GSW06] S. Gajek, J. Schwenk und C. Wegener. SSL-VA-Authentifizierung als Schutz vor Phishing und Pharming. In Jana Dittmann, Hrsg., *SICHERHEIT 2006, Sicherheit - Schutz*

*und Zuverlässigkeit, Beiträge der 3. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V*, Jgg. 77 of *Lect. Notes Inform., Proc.*, 2006.

[Mod07] ModSSL.org. mod_ssl: The Apache Interface to OpenSSL. http://www.modssl.org/, 2007.

[Moz07] Mozilla Foundation. Mozilla Firefox Browser. http://www.mozilla-europe.org/de/products/firefox/, Oktober 2007.

[MT79] Robert Morris und Ken Thompson. Password Security: A Case History. *CACM*, 22(11):594–597, 1979.

[Sch05] J. Schwenk. *Sicherheit und Kryptographie im Internet*, Kapitel "4. WWW-Sicherheit mit SSL". Vieweg, 2. Auflage, 2005.

[WT99] A. Whitten und J. D. Tygar. Why Johnny Can't Encrypt. In *8th USENIX Security Symposium*, 1999.