Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association Suisse des Electriciens, de l'Association des Entreprises électriques suisses

Band: 89 (1998)

Heft: 19

Artikel: Sichere Kommunikation im Internet : eine Einführung in das SSL-Protokoll

Autor: O'Connor, Luke

DOI: https://doi.org/10.5169/seals-902111

Nutzungsbedingungen

Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Siehe Rechtliche Hinweise.

Conditions d'utilisation

L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. Voir Informations légales.

Terms of use

The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. See Legal notice.

Download PDF: 15.10.2024

ETH-Bibliothek Zürich, E-Periodica, https://www.e-periodica.ch

Die grosse Flexibilität des Internet-Protokolls (IP) hat wesentlich zum Erfolg des World Wide Web beigetragen. Der aktuellen Version des Protokolls fehlen jedoch grundlegende Sicherheitsmerkmale, die eine vertrauliche Kommunikation zwischen zwei beliebigen Computern im Netz erlauben würden. Die Firma Netscape hat versucht, diesen Mangel durch die Integration eines zusätzlichen Protokolls zu beheben, ohne die Funktionen des IP-Protokolls einzuschränken. Mittlerweile hat sich dieses sogenannte SSL-Protokoll zum meistgenutzten Verfahren für die verschlüsselte Übertragung vertraulicher Daten im Internet entwickelt.

Sichere Kommunikation im Internet

Eine Einführung in das SSL-Protokoll

■ Luke O'Connor

Einleitung

Derzeit sind etwa zwanzig Millionen Computer an das Internet angeschlossen. Die Kommunikation zwischen all diesen Rechnern wird durch ein gemeinsames Protokoll, das sogenannte TCP/IP oder kurz IP, geregelt. Das IP (Internet-Protokoll) legt fest, wie Informationen auf einem Computer gefunden werden können, wie sie zu formatieren sind und wie sie von einem Computer zum anderen zu transportieren sind. Der IP-Standard unterstützt eine Reihe nützlicher Dienste wie Zeitsignale, Email, Remote File Transfer, Login-Prozesse und – natürlich – das World Wide Web. In der Anfangsphase wurde das Protokoll vor allem durch Kooperationen und De-facto-Standards, die nicht aus offiziellen Standardisierungsverfahren hervorgingen, weiterentwickelt. Dies war ideal für Bildungseinrichtungen, in denen Informationen oft nur einen geringen oder gar keinen kommerziellen Wert besitzen. Es ist jedoch unbrauchbar für geschäftliche Anwender, die höhere Ansprüche an die Geheimhaltung der übertragenen Information stellen. Leider gehörte die Sicherheit der Kommunikation nicht zu den wichtigsten Anforderungen, die an das Internet-Protokoll gestellt wurden. Sie wurde erst aktiv verfolgt, als sich die kommerzielle Nutzung des Internets abzeichnete. Die mangelhafte Datensicherheit in der derzeitigen IP-Version 4 kann man am Aufwand ermessen, der erforderlich ist, um die IP-Version 6 (Version 5 wurde nicht veröffentlicht) mit Sicherheitsfunktionen auszustatten. Für eine weitverbreitete Anwendung muss sich diese Version, von der erste Implementierungen zugänglich sind, aber erst noch durchsetzen.

In einem anderen Ansatz wird versucht, bestehende Protokolle in höheren Kommunikationsschichten mit zusätzlichen Sicherheitsmerkmalen (high layer security services) zu ergänzen. In diesem Ansatz könnten die sehr nützlichen Eigenschaften des Protokolls wie das Adressierungsschema oder das Routing-Verfahren unverändert bleiben.

Die grundlegenden Sicherheitsmerkmale, die erfüllt werden müssen, sind die Authentifizierung der Kommunikationspartner sowie die Integrität und Vertraulichkeit der Information. Die Authentifizierung ermöglicht den Kommunikationsteilnehmern, sich gegenseitig ihrer Identität zu versichern. Auf diese Weise soll verhindert werden, dass Unbefugte in den Besitz vertraulicher Informationen kommen. Die Integrität der Information

Adresse des Autors
Dr. *Luke O'Connor*, IBM-Forschungslaboratorium Zürich, Säumerstrasse 4, 8803 Rüschlikon

ist gewährleistet, wenn die Daten während der Übertragung über einen unsicheren Kanal vor Veränderungen aller Art durch Dritte geschützt sind. Vertraulichkeit schliesslich wird erreicht, wenn Dritte die gesendeten Daten nicht lesen können oder nicht in der Lage sind, die in den Daten enthaltene Information zu erkennen. Authentifizierung, Integrität und Vertraulichkeit sind grundlegende Merkmale sicherer Kommunikation, durch die weitere und aufwendigere Dienste wie zum Beispiel digitale Zahlungsvorgänge erst ermöglicht werden.

Seit geraumer Zeit haben einige Anbieter eigene Implementierungen derartiger Dienste entwickelt. Ein Beispiel ist der GSM-Standard für Mobiltelefone. Leider sind diese Implementierungen viel zu speziell und unflexibel, um mit dem IP vereinbar zu sein. Um zu illustrieren, welche Aufgaben von einer sicheren Internet-Kommunikation zu erfüllen wären, betrachten wir das Beispiel eines Börsengeschäftes zwischen einem Aktienkäufer in Zürich und einem Händler an der New Yorker Börse. Der Zürcher möchte über das Internet Aktien im Wert von 100000 Franken erwerben. Weiter sei angenommen, dass beide Personen zuvor keinen Kontakt miteinander hatten und daher alle Details der Transaktion über das Internet geschickt werden müssen. Solche Transaktionen müssen, da sie in Zukunft zu den wichtigsten Anwendungen im Internet gehören werden, durch Sicherheitserweiterungen geschützt werden. Wie die Datenübertragung im Internet durch das IP, so wird sich auch diese Aufgabe nur durch ein standardisiertes Verfahren lösen lassen. Dabei wird man zu akzeptieren haben, dass die Standardisierung, wie zuvor schon in anderen Fällen (z.B. Open Systems Interconnection [OSI] Standard), durch technische und politische Hürden behindert werden könnte.

In diesem Beitrag werden die Grundlagen des Secure-Socket-Layer(SSL)-Protokolls erläutert [1]. Dieses Protokoll erlaubt den Aufbau eines sicheren Kommunikationskanals zwischen zwei beliebigen, über das Internet verbundenen Partnern. Zurzeit befindet sich die im März 1996 aktualisierte Version SSL V3.0 in Gebrauch. Eine Arbeitsgruppe «Transport Layer Security» des IETF (Internet Engineering Task Force) beschäftigt sich derzeit mit der Standardisierung dieser Version [2].

SSL wurde von der Firma Netscape im Jahre 1994 entwickelt und in den Browser Navigator integriert. Das Protokoll wird inzwischen auch von Firmen wie Apple, DEC, IBM, Microsoft, Novell und

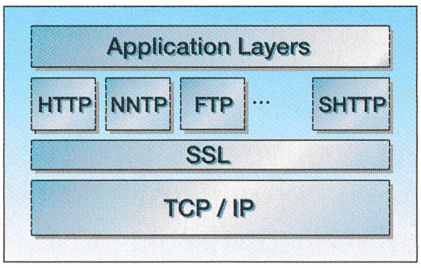


Bild 1 Schichtenmodell des SSL-Protokolls
Die Funktion des Internet-Protokolls TCP/IP bleibt vollständig erhalten.

Sun unterstützt [3]. Netscape schätzt, dass heute etwa 2 Mio. SSL-fähige Browser in Verwendung sind. Ohne Netscapes Entscheidung, das SSL-Protokoll in den eigenen Browser zu übernehmen, hätte es vermutlich mehrere Jahre gedauert, bis sich Anbieter und Forscher auf einen gemeinsamen Standard mit ähnlichen Sicherheitsmerkmalen geeinigt hätten. Netscape hat mit seinem Alleingang vollendete Tatsachen geschaffen und praktisch über Nacht einen neuen Standard etabliert, der seine Bedeutung voraussichtlich für einige Zeit behaupten wird.

Der Name SSL wurde von der ursprünglich in BSD Unix enthaltenen «Standard Socket Networking Library», der ersten Implementierung des Internet-Protokolls, abgeleitet. Der Begriff «layered» bezieht sich dabei auf die Vorgehensweise, in der die Daten für die Übertragung bearbeitet werden. Die Socket Layer und alle tieferliegenden Schichten sind am eigentlichen Übertragungsvorgang beteiligt. Da das SSL-Protokoll in der Socket Layer aktiv ist, können auf höheren Ebenen unterschiedliche Anwendungen (WWW, Remote File Transfer) gesichert werden. Diese Vielfältigkeit ist ein grosser Vorteil des SSL-Protokolls.

Grundlagen der Kryptographie

Um einen ersten Überblick zu gewinnen, betrachten wir einen Client C und einen Server S, die über einen sicheren Kanal miteinander kommunizieren wollen und SSL als Protokoll dieser Kommunikation gewählt haben. (Wir mögen hierbei an den Aktienkäufer in Zürich und den Broker in New York denken.) Aufgabe des Protokolls ist es also, den Kommunikationskanal, der in der SSL-Terminologie als *Session* bezeichnet wird, mit der Möglichkeit zur Authentifizierung, der Integrität und der Vertraulichkeit der Information auszuzeichnen, obwohl zuvor kein Kontakt zwischen Client und Server stattgefunden haben soll.

Die Grundlagen aller sicheren Kommunikationsprotokolle basieren auf der Kryptographie (vom griech. *kryptos* – geheim, *graphein* – schreiben). Der am weitesten verbreitete Verschlüsselungsalgorithmus ist der sogenannte Data Encryption Standard (DES), der vor über 25 Jahren von Forschern der IBM entwickelt wurde. DES wird heute von der Mehrzahl der Bankautomaten verwendet. Der Algorithmus beruht auf einem 56-Bit-Schlüssel K. Mit seiner Hilfe berechnet der Algorithmus aus einem 8-Byte-Eingangswert m einen ebenso grossen Ausgangswert m'. Diese Abbildung wird durch die Gleichung $E_K(m) = m'$ beschrieben, wobei E_K für die Verschlüsselungsoperation (encryption) steht. Diese Operation dient dem Zweck, den Zusammenhang zwischen Aus- und Eingangswert in einer Art zu verschleiern, die eine Berechnung des Wertes von m aus Informationen über m' und über die Funktionsweise des DES-Algorithmus sehr schwierig macht, solange der Schlüssel K nicht bekannt ist. Falls jedoch K zur Verfügung steht, kann die ursprüngliche Nachricht durch die zu E_K inverse Abbildung $m = D_K(m')$ berechnet werden. Mit anderen Worten: Wenn K unbekannt ist, besteht effektiv kein Zusammenhang zwischen m und m'.

Dies ist die Grundlage aller auf Geheimschlüsseln beruhenden Verschlüsselungsmethoden. Wenn sich Client und Server auf einen gemeinsamen Schlüssel und auf DES als Verschlüsselungsalgorithmus geeinigt haben, werden Dritte die Bedeutung der übertragenen Daten nicht erkennen können. Im Fall einer spontanen Transaktion im Internet taucht hier das Problem auf, dass sich die Kommunikationspartner nur des unsicheren Kanals des Internets bedienen können, um die gemeinsamen Geheimschlüssel auszutauschen.

Die Lösung dieses Problems liegt in Verfahren der sogenannten Public-Key-Kryptographie. In diesen Verfahren besitzt jeder Teilnehmer der Kommunikation ein Paar aus einem öffentlichen und einem privaten Schlüssel (K_{pub}, K_{priv}). Der öffentliche Schlüssel dient der Verschlüsselung und der private Schlüssel der Entschlüsselung von Nachrichten. Nehmen wir an, das Schlüsselpaar des Servers heisse (K^S_{pub}, K^S_{priv}). Der Schlüssel K^S_{pub} kann vom Server in einem elektronischen Verzeichnis öffentlich zugänglich gemacht werden. Ein Client, der eine vertrauliche Nachricht an den Server schicken will, muss diesen öffentlichen Schlüssel verwenden, um seine Informationen zu verschlüsseln. Der öffentliche Schlüssel taugt dagegen nicht, um die

Nachricht wieder zu entschlüsseln. Nur der Server kann die verschlüsselte Nachricht mit Hilfe seines privaten Schlüssels K^S_{priv} entschlüsseln. Mit diesem Verfahren erübrigt sich also der Austausch eines geheimen Schlüssels.

Dabei wird vorausgesetzt, dass der öffentliche Schlüssel keine Informationen über den privaten Schlüssel preisgibt, obwohl natürlich zwischen beiden aufgrund ihrer Aufgaben des Ver- und Entschlüsselns ein Zusammenhang bestehen muss. Die Erklärung dieser Eigenschaften erfordert einen gewissen Aufwand an Mathematik, der hier nicht getrieben werden soll. Das folgende Beispiel soll aber einige Ideen der Funktionsweise des sehr verbreiteten RSA-Public-Key-Verfahrens (benannt nach den Erfindern Rivest, Shamir, Adleman) vermitteln. Dazu sei daran erinnert, dass eine Zahl n eine Primzahl ist, wenn sie nur in die Faktoren 1 und sich selbst zerlegt werden kann. Die Folge der Primzahlen beginnt also mit 1, 2, 3, 5, 7, 11, 13. Primzahlen können als die «Atome» der Zahlentheorie aufgefasst werden, denn jede beliebige Zahl kann als Produkt von Primzahlen dargestellt werden ($6 = 2\times3$, $18 = 2\times3\times3$, $91 = 7\times13$). An diesen Beispielen sieht man, dass die Faktorisierung für kleine Zahlen eine einfache Aufgabe ist. Wenn aber eine grosse Zahl $n > 10^{200}$ faktorisiert werden soll, stehen bis heute keine praktisch durchführbaren Verfahren zur Verfügung, mit denen dieses Problem schnell gelöst werden könnte. Im RSA-Kryptosystem wählt man zunächst zwei Primzahlen p und q aus und bildet anschliessend das Produkt $n = p \times q$. Aus p und q wird ein Chiffrierschlüssel e und ein Dechiffrierschlüssel d berechnet. Das Paar (e, n) dient als öffentlicher Schlüssel. Da der Schlüssel d aus den beiden Primzahlen p und q berechnet werden kann, wird man sich fragen müssen, wie schwer die Faktorisierung der veröffentlichten Zahl n zu finden sein wird. Die Antwort ist, dass es bei ausreichend grossen Zahlen selbst mit den schnellsten Computern mehrere Generationen dauern würde, um p und q zu bestimmen. Solange keine andere Möglichkeit gefunden wird, das RSA-Verfahren zu brechen, besteht die Sicherheit von RSA auf dieser Schwierigkeit. Weitere Informationen zum RSA-Algorithmus finden sich im Textkasten auf Seite 18. Auch in anderen Public-Key-Verfahren ist das Brechen der Verschlüsselung mit dem Lösen eines mathematischen Problems äquivalent. Für die Entwicklung eines neuen Verfahrens wählt man meist ein schwer zu lösendes Problem aus und versucht anschliessend einen Algorithmus zu entwickeln, der das Auffinden des privaten Schlüssels mit Hilfe der öffentlich zugänglichen Informationen zu einem äquivalenten Problem macht.

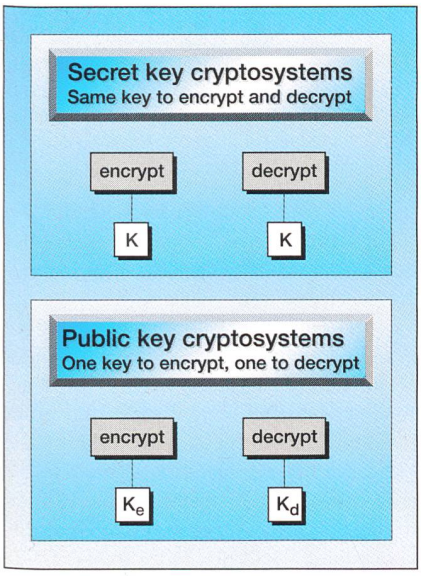
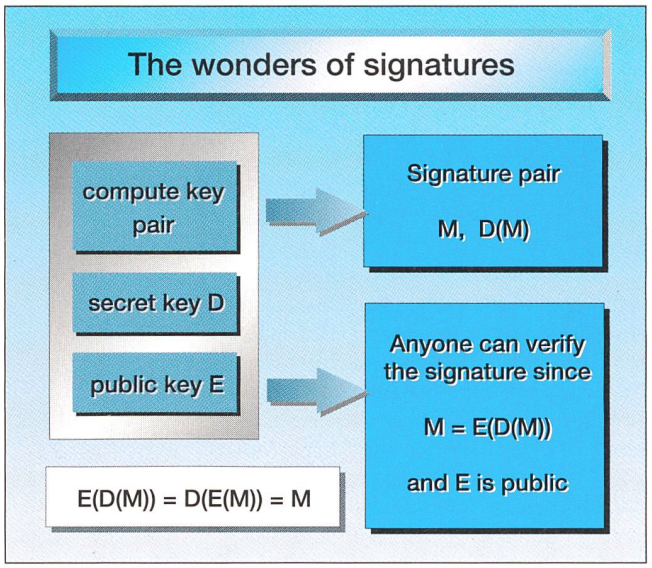


Bild 2 Vergleich zwischen Geheimschlüssel- und Public-Key-Kryptosystemen

Bild 3 Funktionsweise digitaler Unterschriften
In diesem Beispiel wird die ursprüngliche Botschaft M durch Anwenden des Public Keys auf die bereits verschlüsselte Botschaft zurückgewonnen.



Anwendungen von Public Keys

Public-Key-Verfahren wie RSA arbeiten meist sehr viel langsamer als Verfahren mit geheimen Schlüsseln. Die Geschwindigkeitsunterschiede können einen Faktor von 1000 erreichen. Daher liegt es nahe, ein Public-Key-Verfahren nur für den Austausch eines Geheimschlüssels K zu verwenden, der anschliessend zur sicheren und schnellen Übertragung der eigentlichen Daten dienen kann. Der Austausch des Schlüssels K könnte dabei wie folgt vonstatten gehen: Zunächst sendet der Client dem Server eine Zufallszahl R_1, die er mit dem Public Key des Servers verschlüsselt hat. Der Server antwortet durch eine Zufallszahl R_2, die er mit dem Public Key des Client verschlüsselt hat. Nun können beide, Client und Server, den geheimen Schlüssel K beispielsweise als XOR-Funktion der beiden Zufallszahlen definieren: $K = R_1 \oplus R_2$.

Das bedeutendste Einsatzgebiet von Public-Key-Kryptosystemen wird vielleicht die Realisierung digitaler Unterschriften sein. Eine handgeschriebene Unterschrift gilt als Bestätigung, dass der Unterzeichner mit dem Inhalt eines Dokuments einverstanden ist. Da Unterschriften relativ einfach gefälscht werden können, muss ihr Missbrauch durch spezielle Signaturgesetze verhindert werden. Auch digitale Unterschriften sollen das Einverständnis des Unterzeichners bestätigen und zu einem späteren Zeitpunkt nicht verleugnet werden können. Als Beispiel nehmen wir an, der Server erhalte vom Client ein mit dem Public Key des Servers codiertes Dokument D, das er durch seine digitale Unterschrift bestätigen will. Mit Hilfe seines privaten Schlüssels K^S_{priv} bildet er das decodierte Dokument $D' = Sign_S(D)$. Dieser Vorgang ist mit dem Leisten einer handschriftlichen Unterschrift vergleichbar, denn nur der Server ist im Besitz seines privaten Schlüssels, und das Dokument D' kann daher nur von ihm selbst erzeugt worden sein. Der Client erhält vom Server das Paar (D, D') zugeschickt und muss nun noch überprüfen, ob die Codierung von D' mit Hilfe des Public Keys des Servers wieder die ursprünglich gesendete Nachricht D ergibt. Bei dieser Vorgehensweise muss vorausgesetzt werden, dass Codierung und Decodierung inverse und daher vertauschbare Operationen sind. Es spielt dann keine Rolle, ob eine Nachricht zuerst codiert und dann decodiert wird oder

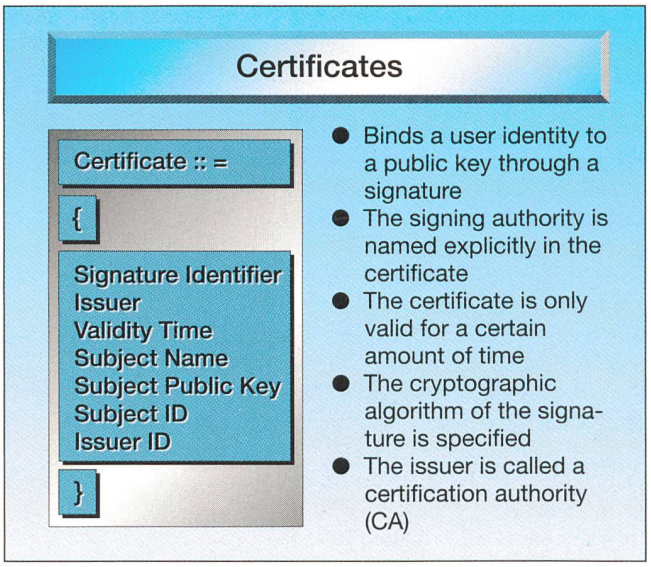
umgekehrt. Das Ergebnis bleibt gleich. Diese Bedingung gilt für das RSA-System, wird aber nicht notwendigerweise von allen Public-Key-Verfahren erfüllt. Allen Verfahren ist jedoch gemein, dass die Unterschrift des Senders mit seinem privaten Schlüssel geleistet wird und durch den Public Key überprüft werden kann.

Da Dokumente sehr gross sein können, wäre die oben skizzierte Unterzeichnungsprozedur häufig zu aufwendig. Man versendet und unterzeichnet daher besser nur einen besonderen Auszug des Dokuments. Hierbei muss gewährleistet sein, dass die Unterzeichnung des Auszugs mit der Unterzeichnung des Originaldokuments gleichwertig ist. Es genügt deshalb nicht, lediglich eine gekürzte Fassung zu versenden. Mit anderen Worten: Jede Veränderung des Originals muss sich auch in einer Änderung des Auszugs niederschlagen. Man benutzt zu diesem Zweck eine kryptographische Hash-Funktion. Eingangswerte dieser Funktion können Dokumente beliebiger Länge sein. Die Ausgangswerte haben immer die gleiche Datenlänge. Die Hash-Funktion hat die Eigenschaft, dass es bei gegebenem Hash-Wert $H(D_1)$ sehr schwierig ist, ein zweites Dokument D_2 zu finden, so dass $H(D_1)=H(D_2)$ gilt. Daher ist die Unterzeichnung des Auszugs $H(D)$ mit einer Signierung des Originaldokuments D gleichwertig.

Die geschilderten Verfahren hängen allesamt von der Verfügbarkeit öffentlicher Schlüssel und ihrer sicheren Zuordnung zu ihrem Besitzer ab. Dieses Problem ist komplizierter, als es die bis hierher geschilderten Beispiele vermuten lassen. Die Schlüssel müssen zunächst in einer gemeinsamen, standardisierten Form repräsentiert werden können. Am häufigsten wird hierzu das sogenannte X.509-Format verwendet, das sich vom X.500 Directory Service ableitet und von der ISO zertifiziert worden ist. Dieses Format definiert ein Objekt, ein sogenanntes Zertifikat, das eine Verbindung zwischen dem Namen des Servers und seinem Public Key herstellt. Ausserdem enthält das Zertifikat die Versionsangaben, Numerierung, die Gültigkeitsdauer und weitere optionale Informationen wie die Email- und die Home-Page-Adresse oder andere Angaben. In Zukunft wird fast jedermann ein eigenes Zertifikat als digitalen Identitätsnachweis besitzen müssen.

Alle Zertifikate werden in einem öffentlichen Verzeichnis, der sogenannten «Public Key Infrastructure» (PKI), gespeichert. Einige Länder sind bereits heute damit beschäftigt, eine eigene nationale PKI aufzubauen. Deren Aufgabe wird sein, Zertifikate zu erstellen und zu verteilen, sie zu widerrufen oder zu löschen. Jede PKI gliedert sich in eine Reihe lokal arbeitender Zertifizierungsinstanzen, die innerhalb einer Stadt, einer Firma oder Universität mit der Verwaltung der Zertifikate betraut sind. Zur Erlangung eines Zertifikats müssen die Teilnehmer ihre Identität zunächst auf klassische Weise durch Vorlegen gewöhnlicher Identitätspapiere (Geburtsurkunden, Pässe usw.) bei der Zerifizierungsbehörde belegen. Die Behörde kann daraufhin das Zerifikat erstellen und unterzeichnen. Damit Zertifikate nicht nur lokal zugänglich sind, müssen zwischen allen Zertifizierungsinstanzen Beziehungen bestehen, die es erlauben, Zertifikate anzufordern und zu überprüfen. Letzteres geschieht durch die Überprüfung der digitalen Unterschrift, die die Zertifizierungsstelle für jedes von ihr erstellte Zertifikat leisten muss.

Bild 4 Inhalt eines Zertifikates



SSL-Details

Mit den bisher geschilderten Grundlagen der Kryptographie können wir nun verstehen, wie SSL einen sicheren Kommunikationskanal mit Authentifizierung, Integrität und Vertraulichkeit aufbaut. Im folgenden werden zunächst die Datenformate beschrieben, und anschliessend wird gezeigt, wie das Handshake-Subprotokoll die kryptographischen Parameter der Session festlegt.

Das SSL-Datensatz-Format

Alle Daten, die während einer Session und zu ihrem Aufbau ausgetauscht werden, befinden sich in speziellen SSL-Datensätzen (SSL-Records). Es gibt drei verschiedene Datensatztypen: SSL-Plaintext, SSL-Compressed und SSL-Ciphertext. Die zu sendenden Daten werden zunächst in SSL-Plaintext-Datensätze geschrieben und anschliessend komprimiert (SSL-Compressed).

Durch Verschlüsselung der komprimierten Datensätze erhält man SSL-Ciphertext-Datensätze, die über das Internet oder einen anderen unsicheren Kanal versendet werden können. Sie bestehen aus den drei folgenden Komponenten:

- Message Authentication Data (MAC)
- Actual Data (*AD*)
- Padding Data (*PD*),

die durch den Ausdruck

MAC[mac-size] || $AD[n]$ || PD[*padding*]

dargestellt werden können. Hierbei ist || eine Byte-Verkettung und $a[m]$ ein Feld von m Bytes. Padding-Daten sind beliebige Daten, die dem Datensatz hinzugefügt werden bis er eine Länge erreicht hat, die einem Vielfachen der Chiffrier-Blocklänge entspricht. Das DES-Verfahren verlangt zum Beispiel eine durch 8 Bit teilbare Datensatzlänge. Die MAC werden berechnet durch

$$MAC = H\{S \,||\, P2 \,||\, H\{S \,||\, P1 \,||\, SQN \,||\, L \,||\, AD\}\} \quad (1)$$

wobei H eine Hash-Funktion (z.B. MD-5-Algorithmus) ist. S ist ein geheimer String, der aus den Session Key berechnet wird und SQN ist eine 32-Bit-Zahl, die mit jeder ausgetauschten Nachricht um eins erhöht wird. Sowohl der Sender als auch der Empfänger besitzen ein Paar solcher Zahlen für gesendete und empfangene Nachrichten. SQN erlaubt die Numerierung der Datensätze modulo 2^{32} (das heisst, mit 2^{32} beginnt die Zählung wieder bei Null). $P1$ und $P2$ sind Auffüllkonstanten (Padding Constants). Die Grösse L definiert die Länge der eigentlichen Daten (AD).

Das SSL-Handshake-Protokoll

Der grösste Teil des SSL-Protokolls beschäftigt sich mit dem Aufbau einer sicheren Session zwischen den beiden Teilnehmern Client und Server. Eine Session wird durch folgende Parameter festgelegt:

- eine Verbindungs-Identifikationsnummer (Verbindungs-ID), die vom Server festgelegt wird. Sie hat eine Länge zwischen 16 und 32 Byte,
- eine Session-Identifikationsnummer (Session-ID), die vom Client gewählt wird und eine Länge von 16 Byte besitzt,
- einen Master-Schlüssel, der vom Client bestimmt wird und sowohl dem Client als auch dem Server dient, um spezielle Sende- und Empfangsschlüssel zu erstellen. Die Länge dieses Schlüssels hängt von der benutzten Ziffernlänge ab[1],
- Informationen für die Authentifizierung des Clients (optional).

Durch die Erfüllung der ersten drei Punkte dieser Liste wird ein sicherer Kommunikationskanal etabliert. Das Protokoll, das diese Aufgabe löst, wird SSL-Handshake-Protokoll (SSLHP) genannt. Falls erwünscht, kann mit diesem Protokoll auch der vierte Punkt, die Authentifizierung des Clients, geregelt werden. Unter der Annahme, dass Client und Server keine frühere Session durchgeführt haben, lässt sich der Ablauf des SSLHP wie folgt darstellen:

1. Der Client sendet dem Server eine beliebige Anfrage und eine Spezifizierung der vom Client unterstützten Geheimschlüssel-Algorithmen.
2. Der Server antwortet mit der Verbindungs-ID, seinem Public-Key-Zertifikat sowie einer editierten Liste der gemeinsamen Geheimschlüssel-Algorithmen.
3. Der Client erstellt den Master-Schlüssel und sendet ihn in verschlüsselter Form an den Server. Zur Verschlüsselung bedient er sich des Public Keys des Servers, der im Server-Zertifikat enthalten war.
4. Ausserdem berechnet der Client mit Hilfe des Master Keys den Sende- und den Empfangsschlüssel. Nun benutzt er den Sendeschlüssel, um dem Server die Verbindungs-ID zu schicken.
5. Der Server entschlüsselt den Master Key, berechnet ebenfalls Sende- und Empfangsschlüssel und überprüft nun, ob die Verbindungs-ID vom Client korrekt verschlüsselt wurde. Dann sendet der Server die ursprüngliche Anfrage (Punkt 1) in verschlüsselter Form an den Client zurück. Hierzu verwendet er den erzeugten Sendeschlüssel.
6. Der Client überprüft, ob die Anfragedaten korrekt sind. Schliesslich erstellt der Server eine neue Session-ID und sendet sie an den Client.

Falls bereits früher eine Session zwischen Client und Server bestanden hat, muss keine neue Identifikationsnummer vergeben werden. In diesem Fall sendet der Client die bereits existierende Nummer bereits in Schritt 1 an den Server. Falls der Server eine Authentifizierung des Clients verlangt, sendet er nach Schritt 4 eine entsprechende Anfrage an den Client, die dieser in verschlüsselter Form an den Server zurückschickt. Dieser Vorgang wird im Abschnitt Client-Authentifizierung näher erläutert.

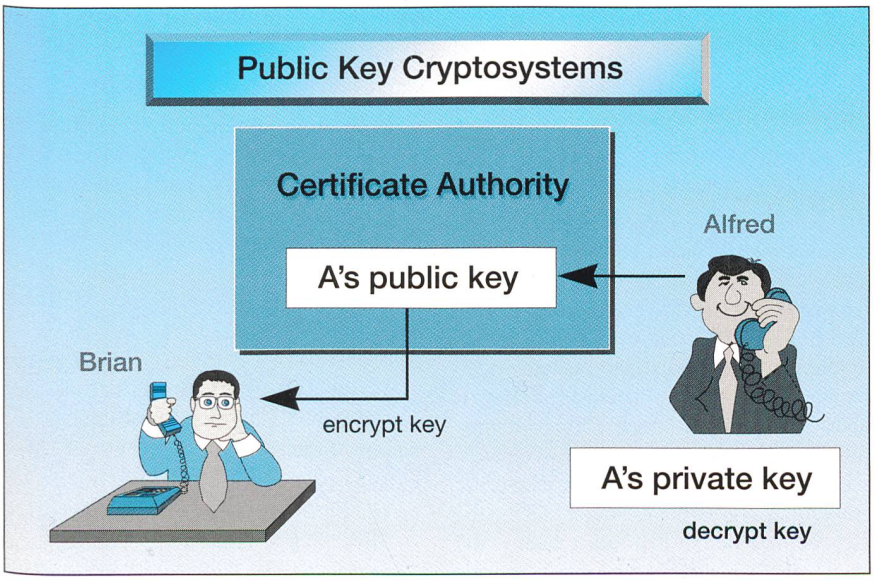


Bild 5 Zertifikate werden von einer Behörde beglaubigt und an die Internet-Nutzer weitergegeben.

Der oben skizzierte Protokoll-Ablauf entspricht der einfachsten Ausgestaltuung des SSLHP und wird in der Spezifikation symbolisch durch die folgenden Ausdrücke dargestellt:

Client-Hello: $C \rightarrow S$: Anfrage, Chiffrier-Spezifikation,
Server-Hello: $S \rightarrow C$: Verbindungs-ID, Server-Zertifikat, Spezifikation gemeinsamer Chiffrier-Algorithmen,
Client-Master-Schlüssel: $C \rightarrow S$: {Master-Schlüssel}Server Public Key,
Client-Ende: $C \rightarrow S$: {Verbindungs-ID} Client-Sendeschlüssel,
Server-Prüfung: $S \rightarrow C$: {Anfrage}Server-Sendeschlüssel,
Server-Ende: $S \rightarrow C$: {Session-ID}Server-Sendeschlüssel.

Die Schreibweise {A}B bedeutet, dass der Dateninhalt von A mit Hilfe des Schlüsseltyps B codiert wird. Alle Nachrichten sind als eine Folge von Bytes mit einer festen Feldanzahl spezifiziert. Die Länge der Felder kann unterschiedlich sein. Als Beispiel betrachte man eine Client-Hello-Nachricht mit zwölf Feldern, die wie folgt definiert seien:

char Msg-Client-Hello
char Client-Version-msb
char Client-Version-lsb
char Länge der Chiffrier-Spezifikationsdaten-msb
char Länge der Chiffrier-Spezifikationsdaten-lsb
char Session-ID-Länge-msb
char Session-ID-Länge-lsb
char Anfragen-Länge-msb
char Anfragen-Länge-lsb
char Chiffrier-Spezifikationsdaten [(msb << 8) || lsb]
char Session-ID-Daten [(msb <<8) || lsb]
char Anfragedaten[(msb << 8) || lsb],

wobei char ein vorzeichenloses (unsigned) Byte und <<8 eine Verschiebung um acht Stellen nach links bezeichnet. Die ersten neun Felder haben eine konstante Länge während die letzten drei eine variable Länge haben. Das erste Feld enthält den Nachrichten-Typ, hier also die Konstante Client-Hello. Im zweiten und dritten Feld wird die vom Client verwendete Versionsnummer des SSL-Protokolls angegeben. Die Versionsnummer ist eine 2-Byte-Grösse, die aus den beiden Bytes msb (most significant byte) und lsb (least significant byte) besteht. Die Längen der Chiffrier-Spezifikationsdaten, der Session-Identifikationsdaten und der Anfragedaten werden durch

[1] Für Sende- und Empfangsschlüssel werden auch die Begriffe Schreib- und Leseschlüssel verwendet.

!¡ª¸È 1/4¸²`!ŸŸ È ¬! `!ŸŸ`!ŸŸÈ¡ª¢C¸¬É MŸ²ŸMÙ¡ª ª ÈŸŸ

Zugegeben, diese Überschrift kann nicht als besonders leserfreundlich bezeichnet werden. Sie stellt aber kein Hindernis dar, das sich nicht mit ein bisschen Einsatz des geneigten Lesers überwinden liesse. Ein Bulletin-Leser mit etwas Geduld und einem guten Taschenrechner wird sie ohne Schwierigkeiten in höchstens vier Stunden Arbeit entschlüsseln können. Die Wahl der einfacheren Variante, die sich am Ende dieses Artikels befindet, sollte nur den wirklich eiligen Lesern vorbehalten bleiben. Hinter der sinnlos erscheinenden Zeichenkette verbirgt sich ein Satz, der mit Hilfe das RSA-Algorithmus verschlüsselt wurde.

Dieses Public-Key-Verfahren beginnt mit der Wahl zweier beliebiger Primzahlen p und q. Diese Zahlen müssen geheim bleiben, denn aus ihnen lassen sich nicht nur der öffentlich zugängliche Chiffrierschlüssel, sondern auch der geheime Dechiffrierschlüssel erzeugen. Als Public Key veröffentlicht der Schlüsselbesitzer das Zahlenpaar (e, n). n wird aus dem Produkt von p und q gebildet und e ist eine beliebige teilerfremde Zahl von $\varphi = (p-1)(q-1)$; φ lässt sich also nicht durch e dividieren. Dies lässt sich an einem einfachen Beispiel illustrieren. Für die Primzahlen p und q soll gelten $p = 11$ und $q = 17$. Das Produkt dieser Zahlen beträgt 187. Als teilerfremde Zahl von $\varphi = (p-1)(q-1) = 160$ können wir $e=3$ wählen. Der öffentliche Schlüssel ist also durch das Paar (3, 187) gegeben. Der Dechiffrierschlüssel d ist definiert als die kleinste natürliche Zahl, die mit e multipliziert und durch φ dividiert den Rest 1 ergibt $(\text{Rest}(ed/\varphi)=1)$. Oder, um es mathematisch auszudrücken: d wird als multiplikatives Inverses mod $(p-1)(q-1)$ von e berechnet: $ed \equiv 1 \pmod{[p-1][q-1]}$

Durch einfaches Ausprobieren kann man in unserem einfachen Beispiel $d = 107$ herausfinden.

Die Überschrift dieses Artikels wurde verschlüsselt, indem zunächst jedem Zeichen des Originalsatzes eine Zahl x zugewiesen wurde, die seiner Stellung im Alphabet entspricht. Das A erhielt also den Wert 1, das B den Wert 2 usw. Anschliessend wurden aus diesen noch unverschlüsselten x-Werten die verschlüsselten y-Werte berechnet. Sie sind als Rest der Division von x^e durch n definiert. Beispiel: $x(\text{«W»}) = 23 \rightarrow x^e = 23^3 = 12\,167 \rightarrow x^e/n = 65$, Rest 12. Also: $y = 12$. Der Rest 12 kann nun als verschlüsselte Botschaft des Buchstabens W versendet werden. Bei der Entschlüsselung der Botschaft geht man in der genau gleichen Weise vor. Man berechnet den Rest, der sich bei der Division von y^d durch n ergibt: $x = \text{Rest}(12^{107}/n)=23$. Damit hat der Empfänger die ursprüngliche, uncodierte Nachricht wiedergefunden. In dieser Art wurde auch obige Überschrift berechnet. Um uns auf die druckbaren Zeichen des Ascii-Codes zu beschränken, haben wir zu jedem y-Wert die Zahl 32 addiert. Das W wird daher nicht durch 12 (dies wäre ein nicht druckbares Steuerzeichen), sondern durch das Zeichen «,» mit dem Ascii-Code 44 dargestellt. Wer sich die Mühe machen will, muss daher von jedem Zeichen der Überschrift den Ascii-Wert bestimmen, die Zahl 32 subtrahieren und x als Rest der Division $12^{107}/n$ bestimmen. *hst*

Executive Summary:
ACHWIEGUTDASSNIEMANDWEISSDASSICHRUMPELSTILZCHENHEISS

Paare von 1 Byte grossen Feldern (Felder 4 bis 9) definiert. Die Länge der Chiffrier-Spezifikationsdaten beispielsweise wird durch den Ausdruck (msb << 8) || lsb festgelegt, wobei msb die Länge der Chiffrier-Spezifikationsdaten-msb und lsb die Länge der Chiffrier-Spezifikationsdaten-lsb bedeutet.

Alle Nachrichten werden im SSL-Protkoll also im Char-Format übertragen. Felder variabler Länge werden dabei durch die 2 Byte grossen msb/lsb-Paare spezifiziert. Sie können daher eine maximale Länge von 2^{16} Bytes besitzen. Darüber hinaus ist im SSL-Handshake-Protokoll auch eine Liste mit Fehlermeldungen definiert, mit denen Protokollverletzungen charakterisiert werden können.

Auswahl des Chiffrier-Algorithmus und Schlüssel-Erzeugung

Die Chiffrier-Spezifikationsdaten definieren eine Anzahl von Hashfunktionen und Chiffrieralgorithmen (von Geheimschlüssel- und von Public-Key-Verfahren), die vom Client oder vom Server unterstützt werden. SSL unterstützt etwa 30 verschiedene Kombinationen aus Hash-Funktionen und Geheimschlüssel-Algorithmen. Die vier wichtigsten sind DES [3], IDEA [4] und die proprietären Algorithmen RC2 und RC4. Die Algorithmen arbeiten entweder im Electronic-Code-Book(ECB)-Modus, in dem jeder Block einzeln verschlüsselt wird, oder im Cipher-Block-Chaining(CBC)-Modus, in dem Abhängigkeiten zwischen den verschlüsselten Blöcken bestehen [5].

Die RC2-, RC4- und IDEA-Algorithmen benutzen 128-Bit-Schlüssel. Da ihr Export unter das Kriegswaffenkontrollgesetz der USA fällt, sind in den Exportversionen nur Schlüssel mit höchstens 40 Bit grossen Schlüsseln erlaubt. DES verschlüsselt die Daten entweder mit einem 56-Bit-Schlüssel (56 Bits plus 8 Parity Bits) oder als Triple-DES in drei Schritten mit jeweils einem anderen 56-Bit-Schlüssel.

In der Client-Hello-Nachricht sendet der Client eine Liste aller von ihm unterstützten Chiffrier-Algorithmen. Der Server streicht aus dieser Liste alle Spezifikationen, die er nicht kennt, und sendet die Liste der gemeinsamen Algorithmen zurück an den Client. Dieser wählt einen Chiffrier-Algorithmus aus, um den Master-Schlüssel zu erstellen, den er dann mit dem Public Key des Servers codiert an den Server versendet. In der Exportversion werden nur 40 Bits des Master-Schlüssels codiert. Die restlichen 88 Bits werden uncodiert übertragen.

Die Sende- und Empfangsschlüssel werden in einer zu (1) ähnlichen Weise durch Hash-Funktionen berechnet. Man beachte, dass der Server, nachdem er die Client-Master-Key-Nachricht erhalten hat, den Master-Key und mit ihm die Sende- und Empfangsschlüssel berechnen kann. Der Sendeschlüssel des Clients entspricht dem Empfangsschlüssel des Servers und umgekehrt. Der Master-Schlüssel wird selbst nie zur Datenübertragung eingesetzt.

Client-Authentifizierung

Falls sich der Client authentifizieren muss, ändert sich das grundlegende SSL-Protokoll durch eine zusätzliche Zertifikats-Abfrage des Servers und die darauffolgende Antwort des Clients:

Client-Hello: $C \rightarrow S$: Anfrage, Chiffrier-Spezifikation,
Server-Hello: $S \rightarrow C$: Verbindungs-ID, Server-Zertifikat, Spezifikation gemeinsamer Chiffrier-Algorithmen,
Client-Master-Schlüssel: $C \rightarrow S$: {Master-Schlüssel}Server Public Key,
Client-Ende: $C \rightarrow S$: {Verbindungs-ID} Client-Sendeschlüssel,
Server-Prüfung: $S \rightarrow C$: {Anfrage}Server-Sendeschlüssel,

Zertifikats-Abfrage: $S \rightarrow C$: {Authentifizierungs-Typ, neue Anfrage}Server-Sendeschlüssel,

Client-Zertifikat: $C \rightarrow S$: {Zertifikatstyp, Zertifikatsdaten}Client-Sendeschlüssel,

Server-Ende: $S \rightarrow C$: {Session-ID}Server-Sendeschlüssel

Die Authentifizierung des Clients wird erreicht, indem der Client Daten mit Hilfe seines privaten Schlüssels signiert und zusammen mit seinem Zertifikat einschliesslich seines Public Key zur Überprüfung an den Server schickt. Wie bereits erwähnt, wird für die Zertifikate das X.509-Format verwendet [6]. Der Client unterzeichnet Daten, die durch Anwenden der MD5-Funktion auf folgende Daten erzeugt werden:

- den Lese- und Schreibschlüssel
- Daten, die der Server in seiner Zertifikats-Anfrage sendet («neue Anfrage»)
- das unterschriebene Server-Zertifikat, das in der Server-Hello-Nachricht enthalten war

Das Resultat wird gemäss dem PKCS#1-Standard für digitale Unterschriften formatiert. Momentan wird nur MD5 als Hashing-Funktion verwendet. Es ist aber möglich, dass der Server in der Zertifikatsabfrage einen anderen Algorithmus vorschlägt.

Zusammenfassung

Wir haben diese Betrachtungen mit dem Problem begonnen, wie das IP-Protokoll durch Sicherheitsmerkmale ergänzt werden kann. Als eine Möglichkeit haben wir das SSL-Protokoll betrachtet, mit dem sichere Kommunikationskanäle zwischen beliebigen Computern im Internet hergestellt werden können. SSL beruht auf Public-Key-Systemen zum Schlüsselaustausch und für digitale Unterschriften. Die Unterschriften dienen der Authentifizierung der Zertifikate der Kommunikationsteilnehmer und damit ihrer sicheren Identifizierung. Im SSL-Protokoll werden alle Daten in speziell formatierten Datensätzen verpackt. Die Integrität der Daten wird durch eine Hash-Funktion garantiert. Die eigentliche Datenübertragung wird mit Hilfe von geheimen Schreib- und Leseschlüsseln, die erst während einer Session erstellt werden, durchgeführt.

Literatur

[1] http://home.netscape.com/newsref/std/SSL.html.

[2] http://www.ietf.org/html.charters/wg-dir.html#SecurityArea.

[3] National Bureau of Standards: Data Encryption Standard. FIPS PUB 46, Washington D.C., January 1977.

[4] *X. Lai:* On the design and security of block ciphers. ETH Series in Information Processing, editor J. Massey, Konstanz, Hartung-Gorre Verlag, 1992.

[5] *C. H. Meyer and S. M. Matyas:* Cryptography: A new dimension in computer security. New York, Wiley, 1982.

[6] CCITT. Recommendation X.509. The Directory – Authentication Framework, 1993. Also referred to as ITU-T Recommendation X.509.

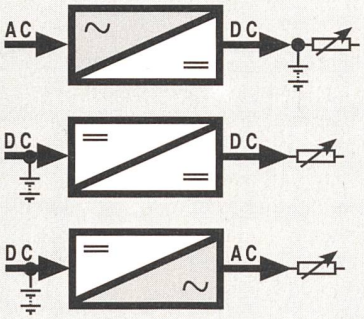
La communication en sécurité sur Internet

Introduction au protocole SSL

La grande facilité du protocole Internet a largement contribué au succès du World Wide Web. Cependant la version actuelle du protocole manque de caractéristiques fondamentales de sécurité permettant une communication confidentielle entre deux ordinateurs quelconques du réseau. La société Netscape a tenté de pallier le défaut en intégrant un protocole supplémentaire sans limiter les fonctions du protocole Internet. Depuis, ce protocole appelé SSL est devenu le procédé le plus utilisé de transmission codée des données confidentielles sur Internet.

Ihr Partner für Batterieladetechnik und gesicherte Stromversorgung

technisch innovativ
breites Sortiment
angepasste Lösungen



BENNING
Power Electronics GmbH
Industriestrasse 6, CH-8305 Dietlikon
Tel. 01 805 75 75, Fax 01 805 75 80
e-mail: benning@point.ch

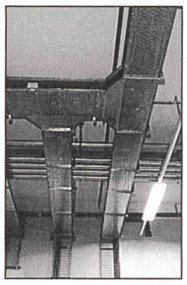
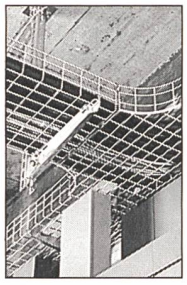
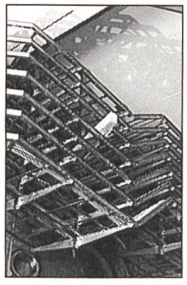
Der Leser ist's

der Ihre Werbung honoriert!

86% der Bulletin-SEV/VSE-Leser sind Elektroingenieure.

91% der Leser haben Einkaufsentscheide zu treffen.

Bulletin SEV/VSE – Werbung auf fruchtbarem Boden.
Tel. 01/448 86 34

LANZ Kabelträgersystem

Multibahnen Kabelbahnen Gitterbahnen Kabelpritschen G-Kanäle Steigleitungen

Das gute und preisgünstige Schweizer Kabelträgersystem aus galv. verzinktem, feuerverzinktem oder rostfreiem Stahl und aus Polyester. Auch farbig.

- Durchdachte Systemteile zur Lösung aller Kabelführungsprobleme.
- neue Verbindungstechnik für rasche Montage
- ohne Wartezeiten sofort lieferbar

Beratung und Angebot von Ihrem Elektrogrossisten u.
lanz oensingen 062/388 21 21 Fax 062/388 24 24

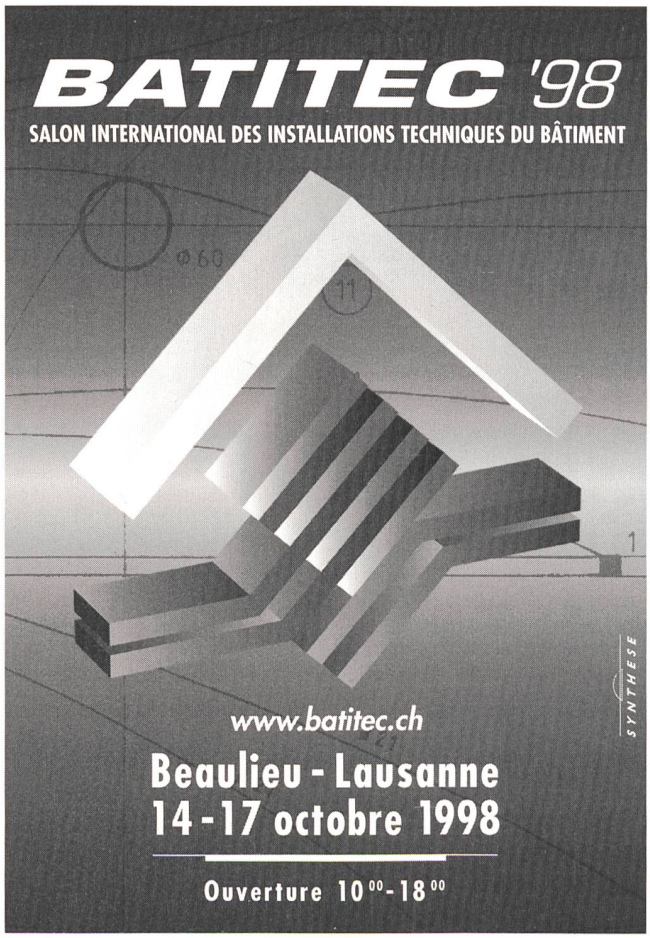
Das LANZ Kabelträgersystem interessiert mich!
Bitte senden Sie Unterlagen über:

☐ LANZ Kabelträgersystem aus galv. Stahl
☐ idem, aus feuerverzinktem Stahl
☐ idem, aus rostfreiem Stahl
☐ LANZ Kabelträgersystem aus Polyester
☐ LANZ G-Kanäle
☐ LANZ Steigleitungen
☐ _____

☐ Könnten Sie mich besuchen? Bitte tel. Voranmeldung!
Name/Adresse/Tel.: _____

10

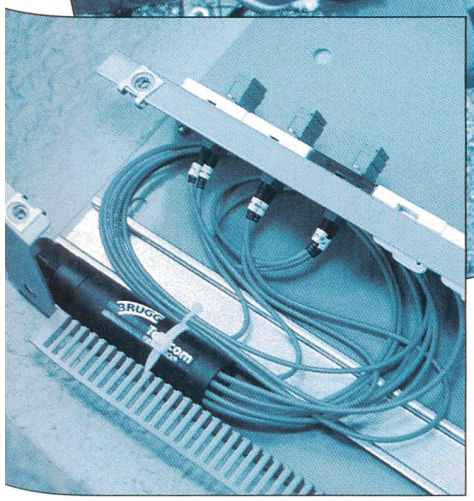
lanz oensingen ag
CH-4702 Oensingen · Telefon 062 388 21 21



Spitzenleistungen in der Übertragungstechnik

«Auf»Schalten zur Zielfahrt

orbit
Basel, 22.–26.9.1998
Halle 300 · Stand B20



Wer in der Formel 1 schon beim Training schnell ist, startet aus der Pole Position. Immer kürzere Zeiten werden auch beim Bau und Betrieb von Kommunikationsnetzen gefordert. Früher standen bei der Installation und Messung von Glasfaserkabeln nicht der Faktor Zeit als vielmehr Spezialkenntnisse und teure Geräte zur Diskussion. Mit FIBER-QUICK® liefern wir Ihnen Kabelverbindungen mit bis zu 48 Glasfasern und fixfertig montierten Steckern an. Sie bestellen einfach die Kabellänge mit der gewünschten Steckerzahl und erhalten von uns eine fertige FIBER-QUICK®-Verbindung. Diese ist nach der Montage sofort betriebsbereit. Zeitaufwendige Spleissarbeiten und Messungen entfallen. Mit FIBER-QUICK® schicken wir Ihnen modernste Technik anschlussfertig franko Domizil. Über kürzere und günstigere Montagezeiten freuen sich nicht nur Ihre Monteure, sondern auch Ihre Kunden. Mit FIBER-QUICK® starten Sie aus der Pole Position und stehen schon kurz nach dem «Auf»-Schalten auf einem guten Podestplatz.

BRUGG Telecom

Brugg Telecom AG · Nachrichtenkabel und Systeme · 5201 Brugg
Telefon 056 46 03 100 · Fax 056 46 03 531 · http://www.bruggtc.ch

Leistung, die verbindet