Kryptographische Protokolle
Hauptseminar im Sommersemester 2009

Protokolldesign

Franz Saller
Technische Universität München

16.06.2009

1 Einleitung

Protokolle werden allgemein definiert als Vereinbarung zwischen Kommunikationspartnern über Art, Inhalt und Formatierung der ausgetauschten Nachrichten, sowie über das Wechselspiel bei der Abwicklung eines Dialogs. Bei kryptographischen Protokollen müssen einzelne Nachrichten chiffriert, dechiffriert oder signiert bzw. verifiziert werden. Grundsätzlich sollte jedes kryptographische Protokoll die Kriterien Geheimhaltung, Integrität, Authentizität, Originalität und Verbindlichkeit gewährleisten.
Sichere und korrekte Protokolle zu entwickeln scheint auf den ersten Blick kein größeres Problem darzustellen, doch bei näherer Betrachtung eröffnet sich ein weites Feld von potentiellen Sicherheitslücken und Angriffspunkten. Um komplexe kryptographische Protokolle, die u.a. zum authentifizierten Schlüsselaustausch dienen, verstehen und analysieren zu können, eignet sich das nach seinen Autoren benannte *Needham-Schroeder-Protokoll* hervorragend. Dieses diente unter anderem dem Kerberos-Authentifizierungsdienst als Ausgangsbasis.
Im Rahmen des Protokolldesigns respektive der Analyse kryptographischer Protkolle wird häufig das klassische *Dolev-Yao Angreifer-Modell* verwendet, bei dem angenommen wird, dass ein Angreifer jede Nachricht mithören, abfangen und manipulieren kann. Einzige Einschränkung bei diesem Modell ist, dass der Angreifer verschlüsselte Nachrichten nur dann entschlüsseln kann, wenn er Kenntnis von dem zur Nachricht passenden Schlüssel hat. Die geheimen Schlüssel der anderen Teilnehmer kennt der Angreifer dabei jedoch nicht, sodass er ihre Nachrichten folglich auch nicht entschlüsseln kann.

2 Das Needham-Schroeder-Protokoll

Message 1: $A \rightarrow S : A, B, N_a$
Message 2: $S \rightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$
Message 3: $A \rightarrow B : \{K_{ab}, A\}_{K_{bs}}$
Message 4: $B \rightarrow A : \{N_b\}_{K_{ab}}$
Message 5: $A \rightarrow B : \{N_b + 1\}_{K_{ab}}$

Grundsätzlich geht es im Needham-Schroeder-Protokoll darum, dass zwei Teilnehmer A und B miteinander kommunizieren wollen. Dazu ist jedoch ein gemeinsamer Schlüssel K_{ab} notwendig, der von einer vertrauenswürdigen Instanz, dem Schlüsselverteilungsserver S, vergeben wird. Im Folgenden soll die in dieser Arbeit verwendete Syntax und Semantik für Protokolle allgemein, sowie die Funktionsweise des Needham-Schroeder-Protokolls erläutert werden:

2.1 Notation

- Die Nachrichtennummer neben **„Message“** gibt nur die vom Protokolldesigner geplante Anordnung der Nachrichten an. Eine einzelne Nach-

richt weiß jedoch nicht, welche Nummer sie hat. Durch technische Fehler oder Angriffe auf die Kommunikation können jedoch einzelne Nachrichten verloren gehen, was bei der Gestaltung der Nachrichtenfolge beachtet werden muss.

- $A \rightarrow B$ bedeutet, dass der Teilnehmer A die Nachricht zu B schickt. A und B können auf diese Information nicht zugreifen, sie müssen den Absender bzw. Empfänger allein anhand des Nachrichteninhalts identifizieren. Die Information dient einzig und allein der Übersicht bzw. der Verständlichkeit beim Protokolldesign.

- **N** ist eine sogenannte *Nonce*, also ein idealerweise ad-hoc gewählter, eindeutiger, einmaliger Identifikator, wie beispielsweise eine zufällig generierte Kombination von Buchstaben oder Zahlen. (*„Number used once“*oder *„only used once“*). Typischerweise werden Nonces eingesetzt, um Replay-Attacken (Attacken, die auf dem Wiedereinspielen von bereits gesandten Nachrichten basieren) oder Man-in-the-Middle Angriffe zu vereiteln. Sie dienen neben den Zeitstempeln (engl. *timestamps*) vorwiegend zur Gewährleistung der Originalität.

- $\{N\}_{K_{as}}$ bedeutet, dass die Nonce N mit dem Schlüssel K_{as} verschlüsselt wurde. Der Schlüssel K_{as} ist dabei nur den Teilnehmern A und S bekannt, die damit auch wieder die Nachricht entschlüsseln und damit die Nonce extrahieren können. In dieser Arbeit werden ausschließlich symmetrische Kryptosysteme gezeigt, bei denen derselbe Schlüssel sowohl zur Verals auch zur Entschlüsselung verwendet wird. Für die Schlüsselgenerierung ist ein vertrauenswürdiger Authentifizierungs- bzw. Schlüsselverteilungsserver notwendig, der im Folgenden als **S** bezeichnet wird. Dieser Server unterliegt strengen Sicherheitsrichtlinien und muss daher vor Manipulation geschützt werden. Natürlich darf er die geheimen Schlüssel nicht an unberechtigte Dritte weiterreichen. Diese vertrauenswürdige, sichere Basis ist beispielsweise im Kerberos-Protokoll angenommen.

- Weiterhin wird bei den Needham-Schroeder-Protokollen vorausgesetzt, dass **jeder** Teilnehmer mit dem Authentifizierungsserver S im Vorfeld bereits einen geheimen Master-Key vereinbart hat, der nur dem Teilnehmer und dem Server bekannt ist. Der Master-Key von A und S ist z.B. K_{as}.

2.2 Funktionsweise

1. A stellt eine Anfrage nach einem Sitzungsschlüssel zu S.

2. S übermittelt A einen Sitzungsschlüssel, auf den nur A zugreifen kann; damit wurde die Authentizität gesichert. Zusätzlich schickt S in der verschlüsselten Nachricht noch eine Nachricht mit demselben Sitzungsschlüssel für B, auf die A nicht zugreifen kann; damit wurde die Manipulation des Sitzungsschlüssels gesichert.

3. A schickt die für B bestimmte Nachricht von S weiter an B. B weiß jetzt jedoch nicht, ob es sich um eine Wiedereinspielung (Replay-Attacke) handelt.

4. B beginnt nun den Handshake mit A, schickt eine Nonce N_b verschlüsselt unter K_{ab} an A und erwartet eine Antwort, bei der die Nonce mittels einer vorher vereinbarten Funktion f (hier $+1$) modifiziert wurde.

5. A schickt B die erwartete Antwort zurück und B kann sicher sein, dass A aktiv ist und ebenfalls den Schlüssel K_{ab} erhalten hat. Die Teilnehmer sind authentifiziert und können nun mit dem ausgetauschten Sitzungsschlüssel kommunizieren.

3 Nonces und ihre Ausprägung

Nonces müssen nicht, trotz der gängigen Meinung, zufällig gewählt werden, sondern können auch als vorhersehbare Buchstaben- oder Zahlenfolge, z.B. in Form eines Zählers, in Erscheinung treten. Die Verwendung der Nonce als Zahl, die ständig

inkrementiert wird, hat auf der einen Seite den Vorteil, dass damit die Neuheit einer Nachricht garantiert werden kann. Auf der anderen Seite könnte ein potentieller Angreifer damit Schaden anrichten, sofern die Nonce nicht geschützt wird. Ein potentielles Angriffsszenario stellt das folgende Zeitsynchronisierungs-Protokoll dar. Dabei stellt ein Teilnehmer A eine Anfrage an einen Zeitserver S und akzeptiert dessen Antwort, sofern sie nach einer bestimmten Zeit nach der Anfrage ankommt. N_a sei in diesem Fall die vorhersehbare Nonce und T_a sei die vom Zeitserver zurückgegebene Zeit.

Message 1: $A \rightarrow S : A, N_a$
Message 2: $S \rightarrow A : \{T_a, N_a\}_{K_{as}}$

Ein Angreifer C könnte exakt die gleiche Nachricht 1 mit einem zukünftigen Wert des Zählers (z.B. N_f) zu einem bestimmten gegenwärtigen Zeitpunkt absenden und die Antwort des Servers speichern. Danach fängt er alle Anfragen von A zu S ab und falls N_a kleiner als N_f ist, leitet er sie unverändert zu S weiter. Sobald N_a gleich N_f ist, gibt er seine vorher gespeicherte Antwort des Servers an A weiter und setzt damit die Systemzeit von A zurück. In diesem Anwendungsszenario sollte die Nonce auf jeden Fall verschlüsselt werden, um eine derartige Replayattacke ausschließen zu können.

4 Zeitstempel

Eine weitere Möglichkeit, um die Neuheit (engl. *freshness*) einer Nachricht zu gewährleisten, ist neben Nonces die Anwendung von Zeitstempeln (engl. *timestamps*), die einem Ereignis einen bestimmten Zeitpunkt zuordnen. Stellen wir uns vor, dass ein Empfänger eine Nachricht nur akzeptiert, wenn sich der Zeitstempel der Nachricht in einem angemessenen Zeitintervall um die lokale Systemzeit des Empfängers befindet. Falls sich die Zeitstempel auf eine absolute Zeit beziehen, dann muss beachtet werden, dass die Differenz zwischen den Systemzeiten verschiedener Teilnehmer viel geringer ist als das erlaubte Alter einer Nachricht. Die Synchronisation der Systemzeit der Teilnehmer bildet unter diesen Voraussetzungen einen essentiellen und kritischen Bestandteil des Protokolls und bedarf einer speziellen Behandlung.

4.1 Auftretende Probleme bei fehlender Zeitsynchronisation

Falls die Zeit bei verschiedenen Teilnehmern in einem Netzwerk unterschiedlich schnell verstreicht, entstehen eine Reihe von Problemen. Diese sollen am Beispiel von Kerberos verdeutlicht werden, können jedoch auch auf beliebige andere Fälle übertragen werden. Falls ein Teilnehmer A mit einem Server S kommunizieren will und über eine im Vergleich zur absoluten Zeit **langsameren** Uhr verfügt, könnte er fälschlicherweise in Wahrheit abgelaufene, zeitabhängige Tickets des Kerberos-Protokolls als gültig erkennen, da seine lokale Systemzeit noch im Gültigkeitszeitraum liegt. Dies führt dazu, dass er länger Antworten bzw. Anfragen von dem jeweiligen Server S entgegennimmt. Für einen Angreifer wird damit das *„Replay-Window“*, das Zeitfenster, in dem er Replay-Attacken durchführen kann, größer. In dieser Zeit nimmt er die Identiät des Servers S an, mit dem A kommunizieren will. Eine Synchronisation der Uhren zwischen Server und den Netzteilnehmern würde dieses Replay-Window erheblich verkleinern.

Die von einer im Vergleich zur absoluten Zeit **schnelleren** Systemzeit eines Teilnehmers ausgehende Gefahr ist weniger intuitiv verständlich und bedarf einer Erläuterung. Angenommen, zum absoluten Zeitpunkt T_0 stellt Benutzer A eine Anfrage nach einem Ticket für die Kommunikation mit dem Server S beim Ticketserver TS, wobei das Ticket bis T_1 gültig sein soll. Für A verstreicht die Zeit nun schneller bis zu T_1 und er akzeptiert das Ticket nur bis $T_{1'}$, wobei $T_{1'} < T_1$. Ein potentieller Angreifer, der Zugriff auf den Rechner von A hat, könnte nun Anfragen von A zum Server S zu einem Zeitpunkt T_X mit $T_{1'} < T_X < T_1$ wiedereinspielen und damit bis zum tatsächlichen Eintreten (nach absoluter Zeitrechnung) von T_1 das Zeitfenster für seine Zwecke nutzen. Er kann in diesem Zeitfenster im Gegensatz zu A dasselbe Ticket weiter für eine Kommunikation mit dem Server S verwenden, schließlich bleibt das

Ticket bis T_1 gültig.
Die offensichtlichste Demonstration der Gefahr in einer nicht-zeitsynchronisierten Kerberos-Umgebung lässt sich mittels eines einfachen Beispiels veranschaulichen. Angenommen, ein Computer startet automatisch neu, weil seine Batterie beschädigt ist. Danach kann angenommen werden, dass seine Systemzeit nicht mehr korrekt funktioniert und er auch die Neuheit von Nachrichten aus dem Netzwerk nicht mehr überprüfen kann. Ein Angreifer könnte diese Situation nun ausnutzen, indem er den Netzwerkverkehr des letzten Tages über eine Replay-Attacke einspielt und damit dem Opferrechner einen alten, eventuell geknackten Sitzungsschlüssel für die Kommunikation unterschiebt. Das Opfer hat nun keine Möglichkeit, den Schlüssel als veraltet zu erkennen.

5 Neuheit von Schlüsseln

Ein weiterer zentraler Aspekt beim Entwurf von kryptographischen Protokollen ist die Gewährleistung der Neuheit (engl. *freshness*) von Schlüsseln. Ein gemeinsamer Schlüssel zur Kommunikation zweier Teilnehmer könnte veraltet und kompromittiert worden sein. Der Zeitpunkt, zu dem ein Schlüssel verwendet wurde, ist dabei kein Anhaltspunkt für dessen Sicherheit und Aktualität. Ein kürzlich verwendeter Schlüssel ist dabei nicht besser als einer, dessen letzte Verwendung schon längere Zeit zurückliegt.
Man kann sich dies am Needham-Schroeder-Protokoll verdeutlichen. Die Nachrichten 4 und 5 dienen nur dazu, Teilnehmer B davon zu überzeugen, dass A anwesend und aktiv ist und nicht, dass der gemeinsam genutzte Schlüssel für die Kommunikation K_{ab} neu und sicher ist. Falls es einem dritten Teilnehmer C gelingen sollte, den Schlüssel K_{ab} zu knacken und die Nachrichten 3 bis 5 abzufangen, kann er die Identität von Teilnehmer A annehmen und mit B kommunizieren. Dies geschieht folgendermaßen:
Message 3': $C \rightarrow B : \{K_{ab}, A\}_{K_{bs}}$
Message 4': $B \rightarrow A : \{N_b'\}_{K_{ab}}$
Message 5': $C \rightarrow B : \{N_b' + 1\}_{K_{ab}}$

In Nachricht 3' glaubt B, dass A mit ihm kommunizieren will und beginnt in der Nachricht 4 mit der Handshake-Phase, um eine potentielle Replay-Attacke zu verhindern. C fängt diese Nachricht erneut ab und kann, sofern ihm die Handshake-Funktion f bekannt ist, die mit K_{ab} verschlüsselte Nachricht 5 abschicken. B glaubt nun, dass er mit A kommuniziert und C kann von nun an selber Nachrichten an B schicken, sowie empfangene Nachrichten von B mit K_{ab} entschlüsseln.
Sofern die privaten Schlüssel K_{as} und K_{bs} sicher sind, kann dieses Problem gelöst werden, indem den Nachrichten 2 und 3 Zeitstempel hinzugefügt werden. Dieser Zeitstempel gibt den Zeitpunkt an, zu dem der Sitzungsschlüssel K_{ab} auf dem Server S generiert wurde.
Damit sieht das geänderte Protokoll folgendermaßen aus:
Message 1": $A \rightarrow S : A, B$
Message 2": $S \rightarrow A : \{B, K_{ab}, T, \{A, K_{ab}, T\}_{K_{bs}}\}_{K_{as}}$
Message 3": $A \rightarrow B : \{K_{ab}, A, T\}_{K_{bs}}$

A und B können verifizieren, dass es sich bei diesen Nachrichten um keine Wiedereinspielungen handelt, indem sie die folgende Gleichung überprüfen:
$|Uhrzeit - T| < \Delta t1 + \Delta t2$
Wobei Uhrzeit die lokale Systemzeit, $\Delta t1$ eine erwartete Diskrepanz zwischen der Serverzeit und der lokalen Uhrzeit und $\Delta t2$ die erwartete Ausbreitungsverzögerungszeit im Netzwerk angibt.

Wie bereits erwähnt, ist die Synchronisierung der Uhren der beteiligten Rechner alles andere als unproblematisch und eine globale Systemzeit zu gewährleisten nicht trivial. Ein manipulierter Zeitdienst oder ein Angriff auf die lokale Systemzeit könnte eine unentdeckte Wiedereinspielung ermöglichen.

6 Vertrauen

Bei der Verwendung von Zeitstempeln ist man zwangsläufig mit einer Vertrauensfrage konfrontiert, bei der man entscheiden muss, ob man den übergebenen Zeitstempel eines anderen Teinehmers in einer Nachricht als korrekt einstufen sollte.

Aber auch allgemein treten zahlreiche Fragestellungen auf, die sich unter anderem mit der Transitivität von Vertrauensbeziehungen befassen. Falls A dem Teilnehmer B in Bezug auf bestimmte Funktionen vertraut, könnte sich eine geänderte Erbringung der Funktionalität auch auf die Sicherheit von A auswirken. Es ist ex ante schwierig bis unmöglich für A herauszufinden, ob sich B „gut“ verhält bzw. ob dessen Soll- mit dessen Istfunktionalität übereinstimmt.
Also sollte man sich beim Protokolldesign genau überlegen, von welchen Vertrauensbeziehungen das Protokoll abhängen soll und warum diese Abhängigkeiten sinnvoll oder notwendig sind. Meist sind die getroffenen Entscheidungen in Bezug auf Vertrauensbeziehungen nicht allein logisch nachvollziehbar, daher sollten die Beweggründe für eine bestimmte Möglichkeit explizit erläutert werden und was gegen andere Alternativen sprechen würde. Beispielsweise ist es im Needham-Schroeder-Protokoll sinnvoll anzunehmen, dass beide Kommunikationspartner A und B einer zentralen Instanz S in Bezug auf die Generierung eines sicheren Sitzungsschlüssels vertrauen. Denn so kann zentral die Qualität des Schlüssels gewährleistet werden, was bei einem dezentralen Ansatz schwierig zu bewerkstelligen gewesen wäre.

7 Schluss

Falls die grundlegenden, in dieser Arbeit beschriebenen Prinzipien bei der Gestaltung kryptographischer Protokolle beachtet werden, können gängige und häufige Fehler vermieden werden. Weiterhin sollte verdeutlicht werden, dass neben einem sicheren Verschlüsselungsverfahren die genaue und sorgfältige Protokollgestaltung durch den Designer essentiell für die spätere Kommunikationssicherheit der Teilnehmer ist.

Literatur

[1] Martin Abadi and Roger Needham. Prudent Engineering Practice for Cryptographic Protocols, June 1993. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.3972&rep=rep1&type=pdf`, [Online; accessed 6-Jun-2009].

[2] Don Davis and Daniel E. Geer. Kerberos security with clocks adrift. Technical report, Massachusetts Institute of Technology, September 1995. `http://www.usenix.org/publications/library/proceedings/security95/full_papers/davis.pdf`, [Online; accessed 11-Jun-2009].

[3] Dorethy E. Denning and Giovanni Maria Sacco. Timestamps in Key Distribution Protocols. *Communication of the ACM*, 24:533–536, June 1981.

[4] Claudia Eckert. *IT-Sicherheit : Konzepte - Verfahren -Protokolle*. Oldenbourg Verlag, 2008.

[5] Klaus-Peter Löhr and Torsten Fink. Kryptographische protokolle, 2002. `http://www.inf.fu-berlin.de/lehre/SS02/sysi/slides/SS-11.pdf`, [Online; accessed 6-Jun-2009].