Protokolldesign

Alexander Aprelkin

16.06.2009

1 Motivation

Ein Kryptographisches Protokoll stellt eine Menge von Regeln dar, die die Abfolge von Nachrichten zwischen den Kommunikationspartnern beschreiben. Hierbei sind manche von diesen Nachrichten teilweise oder ganz verschlüsselt. Kryptographische Protokolle werden eingesetzt, um die Geheimhaltung, Authentizität und die Integrität zu garantieren. Somit dienen sie zur Umsetzung der Kryptographie in der Praxis.

Protokolle können Designfehler enthalten, die das gesamte System trotz einer perfekten Verschlüsselung gegenüber Attacken anfällig machen. Ein berühmtes Beispiel ist der Man-In-The-Middle-Angriff von Lowe 1995 auf das Needham-Schröder-Protokoll zum Schlüsselaustausch bei der asymmetrischen Verschlüsselung, der die Notwendigkeit der Korrektur des Protokolls gezeigt hat.

Weil das Design von robusten Protokollen ganz wichtig ist, wurden viele formalen Methoden entwickelt, die auf mathematischen Konzepten basieren. Beim automatisierten Design mithilfe dieser Methoden werden zwar viele Fehler vermieden, aber diese Anwendung ist sehr zeitaufwändig und wird deswegen nur eingeschränkt eingesetzt.

1995 schlugen M. Abadi und R. Needham 11 informelle Prinzipien vor, die aus der Erfahrung mit der Fehleranalyse von Protokollen hervorgegangen sind und den Einsatz der formalen Methoden vermeiden lassen. Diese Prinzipien stellen informelle Richtlinien dar und sind für die Korrektheit des Protokolls weder hinreichend noch notwendig.

2 Notation

In der Literatur ist die folgende Notation weit verbreitet: A und B repräsentieren die beiden Kommunikationspartner z.B. Alice und Bob. S ist der Trusted Server, dem A und B vertrauen. K_a ist der Schlüssel von A. Bei symmetrischen Verfahren ist ${K_a}^{-1}$ gleich K_a, wobei ${K_a}^{-1}$ den inversen Schlüssel von A darstellt. Für die asymmetrischen Verfahren gilt, dass ${K_a}^{-1}$ den privaten und K_a den öffentlichen Schlüssel repräsentiert. $\{X\}_K$ - ist die Nachricht, die mit dem Schlüssel K verschlüsselt ist. K_{ab} ist der gemeinsame Schlüssel von A und B. N_a ist die Nonce (= Number used only once), die A erzeugt hat. T_a ist der Zeitstempel, ein Beispiel für Nonce, die A generiert hat,

3 Prinzipien

Laut dem ersten Prinzip von Abadi und Needham soll jede Nachricht ihrer echten Bedeutung entsprechen. Die Interpretation der Nachricht darf von ihrem Kontext nicht abhängen. Dieses Prinzip stellt eine Oberklasse weiterer Prinzipien dar und wird am besten am Beispiel des Prinzips 3 erklärt.

Das Prinzip 3 lautet wie folgt: Wenn der Name des Kommunikationspartners für die Bedeutung der Nachricht relevant ist, sollte er in der Nachricht explizit angegeben sein.

Das Needham-Schröder-Protokoll soll zur Schlüsselerzeugung und zum Verbindungsaufbau für die Kommunikation mit dem gemeinsamen Schlüssel zwischen A und B dienen. Der Schlüssel K_{ab} wird dabei durch den Trusted Server S generiert, den nur A direkt anspricht. Anschließend schickt A den von S erhaltenen Schlüssel K_{ab} an B und B bestätigt den Empfang. Das Prinzip 3 wird am Beispiel des leicht modifiziereten Needham-Schröder-Protokolls verdeutlicht.

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

Der Unterschied gegenüber dem Needham-Schröder-Protokoll ist hier der, dass in der Nachricht 2 B nicht mit übertragen wird. Dieser Umstand kann einen Man-In-The-Middle-Angriff ermöglichen, wie folgt:
Der Angreifer C fängt die 1. Nachricht ab und ersetzt B durch C und schickt die Nachricht weiter an S. S generiert den Schlüssel K_{ac} für die Kommunikation zwischen A und C und schickt an A die Nachricht $\{N_a, K_{ac}, K_{ac}, A_{K_{cs}}\}_{K_{as}}$. Wenn A diese Nachricht empfängt, kann es nicht erkennen, dass C die erste Nachricht abgefangen hat, da der Schlüssel K_{ac} keinerlei Information darüber trägt und nur eine Bitsequenz darstellt. In der Nachricht 3 schickt dann A an B den von S generierten Schlüssel für die Kommunikation zwischen A und C. Diese Nachricht fängt C wiederum ab und somit erhält C diesen Schlüssel.
Die letzten 2 Nachrichten dienen nur als Begrüßung zwischen A und C und der Nachweis, dass A und C denselben Schlüssel verwenden. Die gesamte Angriffssituation sieht dann wie folgt aus:

Message 1: $A \rightarrow C : A, B, N_a$
Message 1': $C \rightarrow S : A, C, N_a$
Message 2: $S \rightarrow A : \{N_a, K_{ac}, K_{ac}, A_{K_{cs}}\}_{K_{as}}$
Message 3: $A \rightarrow C : \{K_{ac}, A\}_{K_{cs}}$
Message 4: $C \rightarrow A : \{N_c\}_{K_{ac}}$
Message 5: $A \rightarrow C : \{N_c - 1\}_{K_{ac}}$

Unter Beachtung des Prinzips 3, falls B in der Nachricht 2 direkt angegeben wäre, das heißt wäre man davon ausgegangen, dass der Kontext dieser Nachricht nicht beachtet werden würde, wäre

dieser Angriff nicht möglich. Dies ist glücklicherweise beim echten Needham-Schröder-Protokoll der Fall.

Ein weiteres wichtiges Prinzip bezieht sich auf die Verschlüsselung. Laut diesem Prinzip sollte man sich genau überlegen, warum die Verschlüsselung eingesetzt werden soll und in welchen Fällen es überflüssig sein könnte. Die Gründe für den Einsatz der Verschlüsselung können unter anderem sein: die Vertraulichkeit, die Authentizität oder die Integrität zu garantieren. Wichtig zu erwähnen ist, dass die Verschlüsselung nicht immer verwendet werden soll, wenn sie nicht gerade notwendig ist. Die meisten gängigen Verschlüsselungsverfahren sind nämlich relativ teuer und ihr Einsatz sollte deswegen solange dies die Sicherheit nicht beeinträchtigt, vermieden werden.

Um die möglichen Gründe für den Einsatz der Verschlüsselung zu verdeutlichen, wird das vereinfachte Kerberos-Protokoll betrachtet. Hierbei bedeutet S den Key-Distribution-Server, der das Ticket $\{Ts, L, K_{ab}, A\}_{K_{bs}}$ und den Schlüssel K_{ab} für die Kommunikation zwischen A und B erzeugt. B bedeutet den Server, dessen Dienst A nutzen möchte. L soll die Lebenszeit des erteilten Tickets repräsentieren. Falls der Key-Distribution-Server A auf ihre Anfrage authentifizieren kann, schickt der Key-Distribution-Server das Ticket für die Kommunikation mit B, den von S erzeugten Schlüssel K_{ab}, die Lebenszeit des Tickets, den Namen von B und seinen Zeitstempel. A erhält die Nachricht, entschlüsselt sie mit dem Schlüssel von A und kontrolliert, ob der Name von B mit dem in der ersten Nachricht angegebenen übereinstimmt. In der nächsten Nachricht schickt A dieses Ticket an den Server B, der es nun mit seinem Schlüssel K_{bs} entschlüsselt. Somit erhält B den Schlüssel K_{ab}, mit dessen Hilfe B den zweiten Teil dieser Nachricht entschlüsselt. Falls alle Kontrollen erfolgreich sind, schickt B an A eine Bestätigung an A.

Message 1: $A \rightarrow S : A, B$
Message 2: $S \rightarrow A : \{Ts, L, K_{ab}, B, \{Ts, L, K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$
Message 3: $A \rightarrow B : \{T_s, L, K_{ab}, A\}_{K_{bs}}, \{A, T_a\}_{K_{ab}}$
Message 4: $B \rightarrow A : \{T_a + 1\}_{K_{ab}}$

Wie man leicht erkennen kann, ist die Verschlüsselung in der Nachricht 1 nicht essenziell, da der Key-Distribution-Server nur authentifizierte Benutzer erlaubt und deswegen ein Man-In-The-Middle-Angriff nicht möglich wäre. Allerdings könnte der Angreifer den Key-Distribution-Server mit Nachrichten überfluten, was hier aber als unwahrscheinlich angenommen wird. In der zweiten Nachricht wird Verschlüsselung eingesetzt, um die Authentizität von T, die Geheimhaltung von K_{ab} und die Integrität zu garantieren. In der dritten Nachricht dient die Verschlüsselung mit dem Schlüssel K_{ab} dem Nachweis, dass A den Schlüssel K_{ab} besitzt. Mit demselben Ziel wird die Verschlüsselung in der 4. Nachricht verwendet.

An diesem Beispiel sieht man, dass jeder Einsatz der Verschlüsselung ein Ziel haben muss und dementsprechend nur dann verschlüsselt werden soll, wenn das System ohne die Verschlüsselung nicht sicher funktionieren würde. Sonst wird die Nachricht im Klartext übertragen. Das Engineering-Prinzip KISS “Keep It Simple Stupid“ ist an diesem Prinzip leicht zu verstehen.

Ein weiteres Prinzip gilt als Empfehlung, die Daten erst zu signieren und dann zu verschlüsseln und nicht in der umgekehrten Reihenfolge, da ein Dokument zum Zeitpunkt der Unterzeichnung

dem Unterzeichner klar lesbar sein muss. Die Beachtung dieses Prinzips vermeidet einen einfachen Man-In-The-Middle Angriff, bei dem eine zuerst verschlüsselte und dann signierte Nachricht übertragen wird. Die ursprüngliche Signatur wird entfernt und wird durch die Signatur des Angreifers ersetzt. Dann wird die Nachricht vom Angreifer an den ursprünglichen Empfänger weitergeschickt. Der verschlüsselte Teil der Nachricht blieb unverändert, allerdings stammt er vom ursprünglichen Absender und nicht vom Angreifer, wie der Empfänger denken würde. Dies ist eine Verletzung der Authentizität.

Als ein Beispiel hierzu diene das CCITT Protokoll X.509 One Pass Authentification. Dabei wird mit $Sig()_a$ die Signatur des Unterzeichners A gemeint. Ziel des Protokolls ist eine Authentikation von A gegenüber B. Gleichzeitig wird auch ein von A generierter Schlüssel K_{ab} nach B geheim und authentisch transportiert. Mit X_a werden Nutzerdaten gekennzeichnet, die von A generiert und authentisch transportiert werden.

Message 1: $A \rightarrow B : A, T_a, N_a, B, X_a, Sig_a(T_a, N_a, B, X_a, \{K_{ab}\}_{K_b})$

Der Angreifer verändert infolge eines Man-In-The-Middle-Angriffs die Nachricht wie folgt:

Message 1: $A \rightarrow C : A, T_a, N_a, B, X_a, Sig_a(T_a, N_a, B, X_a, \{K_{ab}\}_{K_b})$
Message 1': $C \rightarrow B : C, T_a, N_a, B, X_a, Sig_c(T_a, N_a, B, X_a, \{K_{ab}\}_{K_b})$

Beim Empfangen der Nachricht 1' denkt B, dass diese Nachricht von C stammt und insbesondere, dass der Schlüssel K_{ab} für die Kommunikation mit C erzeugt wurde. Dies ist aber falsch und wird als Unknown-Key-Share-Attack bezeichnet. Würde man allerdings die Richtlinie beachten, dass man zuerst signieren und dann verschlüsseln sollte, würde dieser gravierende Fehler erst gar nicht auftreten können.

4 Schlussfolgerung

Die hier betrachteten Design-Prinzipien für kryptographische Protokolle können als eine informelle Methode nicht die hinreichende Korrektheit des Protokolls garantieren. Allerdings werden allein durch ihre Einhaltung viele Designfehler vermieden und außerdem wird es möglicherweise auf die teueren formalen Methoden verzichtet. Im Grunde muss man sich beim Protokolldesign im Klaren darüber sein, dass stets die einfachste und vernünftige Lösung gewählt werden sollte.

5 Weiterführende Literatur

- Martin Abadi and Roger Needham "Prudent Engineering Practice for Cryptographic Protocols". 1994
- Martin Abadi „Security Protocols: Principles and Calculi Tutorial Notes". 1994
- Debra S. Herrmann „A Practical Guide to Security Engineering and Information Assurance". Auerbach Publications, 2002