Identifikation

14

In den vorigen Kapiteln wurden zwei wichtige kryptographische Basismechanismen erklärt: Verschlüsselung und digitale Signatur. In diesem Kapitel geht es um eine dritte grundlegende Technik: die Identifikation.

14.1 Anwendungen

Wir geben zuerst zwei Beispiele für Situationen, in denen Identifikation nötig ist:

Beispiel 14.1 Alice will über das Internet von ihrer Bank erfahren, wieviel Geld noch auf ihrem Konto ist. Dafür muss sie sich der Bank gegenüber identifizieren. Sie muss also beweisen, dass tatsächlich Alice diese Anfrage stellt und nicht ein Betrüger.

Beispiel 14.2 Bob arbeitet in einer Universität und verwendet dort das Computernetzwerk. Bob steht im Benutzer-Verzeichnis des Rechenzentrums der Universität. Wenn Bob morgens zur Arbeit kommt, muss er sich im Netz anmelden und dabei beweisen, dass er tatsächlich Bob ist. Das Benutzermanagement verifiziert, dass Bob ein legitimer Benutzer ist und gewährt ihm Zugang. Als Identifikationsverfahren wird meistens Benutzername und Passwort verwendet. Wir werden diese Identifikationsmethode und ihre Probleme in Abschn. 14.2 diskutieren.

Identifikation ist in vielen Anwendungen nötig. Typischerweise geht es dabei um die Überprüfung einer Zugangsberechtigung, die an eine bestimmte Identität gebunden ist. Verfahren, die Identifikation ermöglichen, nennt man *Identifikationsprotokolle*. In einem Identifikationsprotokoll beweist ein *Beweiser* Bob der *Verifiziererin* Alice, dass sie tatsächlich mit dem Bob kommuniziert.

Bei der Verwendung von Identifikationsprotokollen tritt allerdings ein Problem auf. Die Verifiziererin Alice kann sich nur davon überzeugen, dass sie im Augenblick des

J. Buchmann, *Einführung in die Kryptographie*, Springer-Lehrbuch,
DOI 10.1007/978-3-642-39775-2_14

erfolgreichen Beweises mit Bob kommuniziert. Sie weiß aber nicht, dass ihr Kommunikationspartner danach immer noch Bob ist. Darum werden in der Praxis komplexere Protokolle verwendet die einen *sicheren Kanal* aufbauen. Ein solcher Kanal verbindet zwei Parteien, die sich gegenseitig identifiziert haben und verhindert, dass Angreifer den Kanal übernehmen. Das bekannteste solche Protokoll ist das *Transport-Layer-Security-Protokoll (TLS)*. Auch bei der Verwendung von Protokollen zum Aufbau sicherer Kanäle spielt Identifikation eine wichtige Rolle und zwar sowohl im Protokoll selbst als auch nachdem der Kanal aufgebaut wurde. So wird in fast allen E-Commerce und E-Banking-Anwendungen zunächst ein sicherer TLS-Kanal aufgebaut. Anschließend identifiziert sich die Nutzerin oder der Nutzer mit Benutzername und Passwort.

In diesem Kapitel werden verschiedene Identifikationsprotokolle besprochen.

14.2 Passwörter

Die Identifikation von Nutzerinnen und Nutzern von Web-Diensten (E-Commerce, E-Banking etc.) funktioniert normalerweise mit Benutzername und Passwort. Das Passwort w wird vom Benutzer ausgewählt und ist nur ihm bekannt. Im Rechner wird der Funktionswert $f(w)$ von w unter einer Einwegfunktion f gespeichert. Wenn der Benutzer Zugang wünscht, gibt er seinen Namen und sein Passwort ein. Der Web-Dienst bestimmt den Funktionswert $f(w)$ und vergleicht ihn mit dem Wert, der zusammen mit dem Benutzernamen in der Liste gespeichert ist. Wenn die beiden Werte übereinstimmen, erhält der Benutzer Zugang, andernfalls nicht.

Die Identifikation mit Benutzername und Passwort führt zu Sicherheitsproblemen. Ein erstes Problem besteht darin, dass sich der Benutzer sein Passwort merken muss. Also muss er das Passwort so wählen, dass er es nicht vergisst. So werden zum beispiel häufig die Namen von Verwandten als Passwörter benutzt. Das eröffnet aber die Möglichkeit eines *Wörterbuchangriffs*: Der Angreifer besorgt sich die Datenbank mit den Bildern $f(w)$ aller Passwörter unter der Einwegfunktion f. Der Angreifer bildet für alle Wörter w aus einem Wörterbuch, in dem wahsrcheinliche Passwörter stehen, den Funktionswert $f(w)$. Er vergleicht diese Funktionswerte mit den abgespeicherten Funktionswerten. Sobald ein berechneter mit einem abgespeicherten Wert übereinstimmt, hat der Angreifer ein geheimes Passwort gefunden. Um diesen Angriff zu verhindern, muss man Passwörter wählen, die nicht in Wörterbüchern vorkommen. Es empfiehlt sich, Sonderzeichen wie z. B. $ oder # in den Passwörtern zu verwenden. Damit wird es aber schwieriger, sich die Passwörter zu merken.

Ein Angreifer kann auch versuchen, das geheime Passwort durch Abhören des Kanals zu finden, durch den das Passwort zum Web-Dienst gesendet wird. Gelingt ihm dies, so kennt er das Passwort und kann sich danach selbst erfolgreich anmelden. Dieser Angriff wird verhindert, indem der Nutzer oder die Nutzerinnen zuerst mit dem Dienst einen sicheren TLS-Kanal aufbauen. Dies ist bei Web-Diensten üblich.

Schließlich kann der Angreifer versuchen, das Passwortverzeichnis zu ändern und sich selbst in das Passwortverzeichnis einzutragen. Das Passwortverzeichnis muss also sicher vor unberechtigten Schreibzugriffen geschützt werden.

14.3 Einmal-Passwörter

Im vorigen Abschnitt wurde gezeigt, dass die Verwendung von Passwörtern problematisch ist. Sicherer sind Einmalpasswörter. Ein Einmalpasswort wird nämlich nur für einen Anmeldevorgang benutzt und danach nicht mehr.

Die einfachste Methode, das zu realisieren, besteht darin, dass der Beweiser eine Liste $w_1, w_2, \ldots, w_n$ geheimer Passwörter und der Verifizierer die Liste der Funktionswerte $f(w_1), \ldots, f(w_n)$ hat. Zur Identifikation werden dann der Reihe nach die Passwörter $w_1, w_2, \ldots, w_n$ benutzt. Die Schwierigkeit dieses Verfahrens besteht darin, dass der Beweiser alle Passwörter schon von vornherein kennen muss. Daher können sie auch vorher ausspioniert werden.

Eine Alternative besteht darin, dass Beweiser und Verifizieren die Kenntnis einer geheimen Funktion f und eines Startwertes w teilen. Die Passwörter sind $w_i = f^i(w)$, $i \geq 0$. Bei diesem Verfahren müssen Beweiser und Verifizierer keine Passwörter a priori kennen. Das Passwort w_i kann als $w_i = f(w_{i-1})$ auf beiden Seiten zum Zeitpunkt der Identifikation berechnet werden. Die geheime Einwegfunktion muss aber wirklich geheim bleibt.

14.4 Challenge-Response-Identifikation

Bei den bis jetzt beschriebenen Identifikationsverfahren kann ein Angreifer lange vor seinem Betrugsversuch in den Besitz eines geheimen Passworts kommen, das er dann zu irgendeinem späteren Zeitpunkt einsetzen kann. Das funktioniert sogar bei Einmalpasswörtern.

Bei Challenge-Response-Verfahren ist das anders. Will sich Alice Bob gegenüber identifizieren, bekommt sie von Bob eine Aufgabe (Challenge). Die Aufgabe kann sie nur lösen, weil sie ein Geheimnis kennt. Die Lösung schickt sie als Antwort (Response) an Bob. Bob verifiziert die Lösung und erkennt die Identität von Alice an, wenn die Lösung korrekt ist. Andernfalls erkennt er Alices Identität nicht an. Da die zu lösende Aufgabe bei jedem Identifikationsvorgang zufällig erzeugt wird, können sich Angreifer darauf nicht so leicht vorbereiten.

14.4.1 Verwendung von Public-Key-Kryptographie

Challenge-Response-Protokolle kann man unter Verwendung von Signaturverfahren realisieren. Will sich Alice Bob gegenüber identifizieren, erhält sie von Bob eine Zufallszahl,

die Challenge, und signiert diese. Sie schickt die Signatur an Bob (Response). Bob verifiziert die Signatur. Ist die Signatur gültig, ist die Identität von Alice bewiesen.

14.4.2 Zero-Knowledge-Beweise

Auch *Zero-Knowledge-Protokolle* erlauben die Identifikation. In einem solchen Protokoll kennt der *Beweiser* ein Geheimnis. Er kann jeden Verifizierer davon überzeugen, dass er das Geheimnis kennt und sich so identifizieren. Der Beweis verläuft dabei so, dass niemand etwas über das Geheimnis lernt. Daher der Name Zero-Knowldege-Protokoll.

Im Folgenden erläutern wir als Beispiel das *Fiat-Shamir-Identifikationsverfahren*, das 1986 von Amos Fiat und Adi Shamir entwickelt wurde (siehe [29]).

Bob, der Beweiser, wählt wie im RSA-Verfahren zwei Primzahlen p und q und berechnet $n = pq$. Dann wählt er eine Zahl s zufällig und gleichverteilt aus der Menge der zu n primen Zahlen in $\mathbb{Z}_n$ und berechnet $v = s^2 \bmod n$. Bobs öffentlicher Schlüssel ist (v, n). Sein geheimer Schlüssel ist s, eine Quadratwurzel von $v \bmod n$. Im Fiat-Shamir-Protokoll beweist Bob der Verifiziererin Alice, dass er eine Quadratwurzel s von v kennt. Wir haben in Abschn. 8.4.5 bewiesen, dass Angreifer, die eine solche Quadratwurzel berechnen können, auch dazu in der Lage sind, n zu faktorisieren. Nach heutiger Kenntnis ist es also unmöglich, das Geheimnis zu finden, wenn n groß genug ist.

1. **Commitment** Bob wählt zufällig und gleichverteilt $r \in \mathbb{Z}_n$. Er berechnet $x = r^2 \bmod n$. Das Ergebnis x sendet Bob an die Verifiziererin Alice.
2. **Challenge** Alice wählt zufällig und gleichverteilt eine Zahl $e \in \mathbb{Z}_2$ und sendet sie an Bob.
3. **Response** Bob berechnet $y = rs^e \bmod n$ und schickt Alice den Wert y.
4. Alice akzeptiert den Beweis, wenn $y^2 = xv^e \bmod n$ ist.

Beispiel 14.3 Es sei $n = 391 = 17 * 23$. Der geheime Schlüssel von Bob ist $s = 123$. Der öffentliche Schlüssel von Bob ist also $(271, 391)$. In einem Identifikationsprotokoll will Bob Alice beweisen, dass er eine Quadratwurzel von $v \bmod n$ kennt. Er wählt $r = 271$. $s = 1$ und schickt $x = r^2 \bmod n = 324$ an Alice. Alice schickt die Challenge $a = 1$ an Bob. Er schickt die Response $y = rs \bmod n = 98$ zurück. Alice verifiziert $220 = y^2 \equiv vx \bmod n$.

Zero-Knowledge-Protokolle müssen drei Eigenschaften haben.

- *Completeness*: Kennt der Beweiser Bob das Geheimnis, akzeptiert die Verfiziererin Alice immer seine Antworten als richtig.
- *Soundness*: Kennt der Beweiser Bob das Geheimnis nicht, so lehnt Alice den Beweis mit Wahrscheinlichkeit $\varepsilon > 0$ ab, selbst wenn der Beweiser sich nicht an das Protokoll hält.

- *Zero-Knowledge-Property*: Die Verfiziererin lernt aus dem Protokoll nichts, insbesondere über das Geheimnis, selbst wenn sie sich nicht an das Protokoll hält. Dies wird im Beispiel formaler gefasst.

Die Soundness-Definition erlaubt, dass Bob vom Protokoll abweichen darf. Betrügerische Beweiser ändern nämlich das Protokoll, wenn ihnen das hilft, erfolgreich zu sein. Entsprechend erlaubt die Zero-Knowledge-Definition, dass die Verifizierin Alice vom Protokoll abweicht. Will Alice nämlich etwas über das Geheimnis lernen und hilft ihr dabei eine Veränderung des Protokolls, so tut sie dies natürlich.

Ist die Wahrscheinlichkeit ε dafür, dass der Beweis eines betrügerischen Beweisers abgelehnt wird, kleiner als 1, muss das Protokoll wiederholt ausgeführt werden, um den nötigen Grad an Sicherheit für eine erfolgreiche Identifikation zu erhalten. Die Wahrscheinlichkeit dafür, dass der Betrüger k-mal erfolgreich ist, beträgt nämlich $(1-\varepsilon)^k$. Diese Wahrscheinlich konvergiert gegen 0.

Wir erläutern nun am Beispiel des Feige-Fiat-Shamir-Protokolls, was diese Eigenschaften bedeuten und wie sie bewiesen werden.

Der Beweis der Completeness ist leicht. Wenn Bob nämlich das Geheimnis, also die Quadratwurzel s aus $v \bmod n$, kennt, so kann er beide möglichen Challenges von Alice so beantworten, dass Alice akzeptiert.

Als nächstes untersuchen wir die Soundness des Protokolls. Angenommen, Bob kann beide Challenges von Alice richtig beantworten. Dann kann er für $e = 0$ und $e = 1$ Zahlen $y_e \in \mathbb{Z}_n$ berechnen mit $y_e^2 \equiv bxv^e \bmod n$. Also kann er $y = y_1 y_0^{-1} \bmod n$ berechnen mit $y^2 \equiv \pm v \bmod n$. Er kennt also das Geheimnis. Dies impliziert, dass Beweiser, die das Geheimnis nicht kennen, eine der beiden Challenges nicht beantworten können. Da die Challenges vom Verifzierer zufällig gleichverteilt gewählt werden, werden also die Beweise von betrügerischen Beweisern, die das Geheimnis nicht kennen, mit Wahrscheinlichkeit mindestens $1/2$ abgelehnt.

Schließlich zeigen wir die Zero-Knowledge-Eigenschaft des Protokolls, dass Alice also nichts aus dem Protokoll lernt, insbesondere über das Geheimnis. Das gilt selbst dann, wenn sich Alice nicht an das Protokoll hält. Dies wird folgendermaßen formalisiert. Wenn Alice etwas aus dem Protokoll lernen möchte, muss sie es aus dem lernen, was der Beweiser kommuniziert. Das *Transkript* der Kommunikation zwischen Beweiser Bob und Verfizierin Alice ist $T = (x, a, y) \in \mathbb{Z}_n \times \mathbb{Z}_2 \times \mathbb{Z}_n$. Es unterliegt einer Wahrscheinlichkeitsverteilung, weil Bob und Alice Werte zufällig wählen. Wir zeigen nun, dass Alice auch ohne Kenntnis des Geheimnisses ein Transkript mit derselben Wahrscheinlichkeitsverteilung *simulieren* kann. Das zeigt dann, dass sie nichts aus dem Protokoll lernt.

Wir erklären die Simulation. Im Feige-Fiat-Shamir-Protokoll entsteht in jedem Durchlauf das Transkript (x, e, y). Dabei ist x ein gleichverteilt zufälliges Quadrat modulo n in $\mathbb{Z}_n$. Außerdem ist e eine Zahl in $\mathbb{Z}_2$, die Alice wählt und die von x abhängen darf. Man beachte, dass Alice sich nicht an das Protokoll halten muss sondern e nach eigenen Regeln wählen darf. Schließlich ist y eine gleichverteilt zufällige Quadratwurzel von xv^e modulo n.

Tripel (x, e, y) mit derselben Verteilung kann ein Simulator ohne Kenntnis einer Quadratwurzel von v modulo n auf folgende Weise erzeugen. Er wählt gleichverteilt zufällige Zahlen $f \in \mathbb{Z}_2$ und $y \in \mathbb{Z}_n$. Dann berechnet er $x = y^2 v^{-f} \bmod n$, wobei v^{-f} das Inverse von v^f modulo n ist. Schließlich wählt der Simulator gemäß der von Alice ausgesuchten Verteilung eine Zahl e in $\mathbb{Z}_2$. Wenn e und f übereinstimmen, gibt der Simulator das Tripel (x, e, y) aus. Andernfalls verwirft der Simulator das Tripel (x, e, y).

Die erfolgreichen Ausgaben des Simulators unterliegen tatsächlich derselben Wahrscheinlichkeitsverteilung wie das Transkript des Originalprotokolls. Es ist nämlich x ein gleichverteilt zufälliges Quadrat modulo n, y eine gleichverteilt zufällige Quadratwurzel von v modulo n und e ein gemäß der Strategie von Alice gewähltes Bit. Die Erfolgswahrscheinlichkeit für den Simulator ist $1/2$.

Die beschriebene Modelierung von Zero-Knowledge-Protokollen erlaubt zahlreiche Verallgemeinerungen. Im Beispiel des Fiat-Shamir-Protokolls sind die vom Simulator erzeugten Verteilungen identisch mit der im Protokoll erzeugten Verteilung. Ein solches Protokoll heißt *perfektes Zero-Knowledge-Protokoll*. Diese Forderung kann abgeschwächt werden. Bei *Statistischen Zero-Knowledge-Beweisen* ist es erlaubt, dass die Verteilungen eine vernachlässigbaren Unterschied haben.

Im allgemeinen fordert man für die Zero-Knowledge-Eigenschaft nur, dass die Verteilung auf den Nachrichten im simulierten Protokoll von der Verteilung der Nachrichten im Originalprotokoll in Polynomzeit nicht zu unterscheiden ist. Dies wird im Rahmen der Komplexitätstheorie mathematisch präzisiert. Wir verweisen auf [31].

14.5 Übungen

Übung 14.1 Sei p eine Primzahl, g eine Primitivwurzel mod p, $a \in \{0, 1, \ldots, p-2\}$ und $A = g^a \bmod p$. Beschreiben Sie einen Zero-Knowledge-Beweis dafür, dass Alice den diskreten Logarithmus a von A mod p zur Basis g kennt. Der ZK-Beweis ist analog zum dem in Abschn. 14.4.2 beschriebenen.

Übung 14.2 Im Fiat-Shamir-Verfahren sei $n = 143$, $v = 82$, $x = 53$ und $e = 1$. Bestimmen Sie eine gültige Antwort, die die Kenntnis einer Quadratwurzel von v mod n beweist.

Übung 14.3 (Feige-Fiat-Shamir-Protokoll) Eine Weiterentwicklung des Fiat-Shamir-Verfahrens ist das Feige-Fiat-Shamir-Protokoll. Eine vereinfachte Version sieht so aus: Alice benutzt einen RSA-Modul n. Sie wählt $s_1, \ldots, s_k$ gleichverteilt zufällig aus $\{1, \ldots, n-1\}^k$ und berechnet $v_i = s_i^2 \bmod n$, $1 \le i \le k$. Ihr öffentlicher Schlüssel ist $(n, v_1, \ldots, v_k)$. Ihr geheimer Schlüssel ist $(s_1, \ldots, s_k)$. Um Bob von ihrer Identität zu überzeugen, wählt Alice $r \in \{1, \ldots, n-1\}$ zufällig, berechnet $x = r^2 \bmod n$ und schickt x an Bob. Bob wählt gleichverteilt zufällig $(e_1, \ldots, e_k) \in \{0, 1\}^k$ und schickt diesen Vektor an Alice (Challenge). Alice schickt $y = r \prod_{i=1}^{k} s_i^{e_i}$ an Bob (Response). Bob verifiziert

$y^2 \equiv x \prod_{i=1}^{k} v_i^{e_i} \bmod p$. Mit welcher Wahrscheinlichkeit kann ein Betrüger in einem Durchgang dieses Protokolls betrügen?

Übung 14.4 Modifizieren Sie das Verfahren aus Übung 14.3 so, dass seine Sicherheit auf der Schwierigkeit beruht, diskrete Logarithmen zu berechnen.

Übung 14.5 (Signatur aus Identifikation) Machen Sie aus dem Protokoll aus Übung 14.3 ein Signatur-Verfahren. Die Idee besteht darin, bei Signatur der Nachricht m die Challenge $(e_1, \ldots, e_k)$ durch den Hashwert $h(x \circ m)$ zu ersetzen.