Nachrichten sicher verschicken

Gruppenaufgabe Tag 2

Das sichere Verschicken von Nachrichten beruht auf Verschlüsselung. Man nimmt eine Nachricht m und einen Schlüssel k und berechnet daraus die Kodierung $E_k(m)$ (Encrypt m with k). Dabei sollte sich $E_k(m)$ unter Kenntnis von k leicht berechnen lassen, m sollte sich leicht aus $E_k(m)$ unter Kenntnis von k berechnen lassen, aber ohne die Kenntnis von k soll die Berechnung von m aus $E_k(m)$ schwierig sein. Selbst wenn das konkrete Verschlüsselungsverfahren aus zwei Schlüsseln besteht, etwa wie bei PGP aus einem öffentlichen und einem geheimen, sprechen wir in dieser Aufgabe nur von einem Schlüssel und setzen eine korrekte Benutzung voraus. Die Sicherheit aktueller Verschlüsselungsmethoden beruht unter anderem auf den Eigenschaften der Faktorisierung großer Zahlen. Für zwei große Primzahlen p, q läßt sich $p \cdot q$ leicht berechnen, aber ohne Kenntnis von p, q läßt sich das Produkt $p \cdot q$ nicht schnell faktorisieren (also in die Faktoren p, q zerlegen). Abhängig von der Größe von p, q und der aktuellen Leistungsfähigkeit von Computern kann man recht gut abschätzen, wie lange es dauern würde, das Produkt $p \cdot q$ zu faktorisieren. Aktuell empfiehlt die Bundesnetzagentur für das Verschlüsseln von E-Mails eine Mindestgröße des Schlüssels von 4096 Bits.

Aufgabe 1: Die Anwendersicht

Diskutieren Sie aus Sicht der Anforderungen des Anwenders einer Verschlüsselungstechnologie wichtige Eigenschaften des Verschlüsselungsverfahrens.

Exkurs: Austausch von Schlüsseln

Ein weiterer wichtiger Aspekt des sicheren Verschickens von Nachrichten ist die Etablierung eines neuen Schlüssels auf Basis bereits vorhandener Schlüssel. Dazu gibt es eine ganze Reihe von Protokollen, also Algorithmen, die auf dem Austausch von Nachrichten basieren. Als Beispiel aus den 1990ern betrachten wir das Protokoll zwischen Alice (A), Bob (B) und einem vertrauenswürdigen Server (T).

$$
\begin{array}{ll}
(1) & A \to B : A, N_A \\
(2) & B \to T : B, E_{k_{\mathrm{BT}}}(A, N_A, T_B), N_B \\
(3) & T \to A : E_{k_{\mathrm{AT}}}(B, N_A, k_{\mathrm{AB}}, T_B), E_{k_{\mathrm{BT}}}(A, k_{\mathrm{AB}}, T_B), N_B \\
(4) & A \to B : E_{k_{\mathrm{BT}}}(A, k_{\mathrm{AB}}, T_B), E_{k_{\mathrm{AB}}}(N_B)
\end{array}
$$

Dabei bedeutet $E_k(m)$ wie oben die Verschlüsselung der Nachricht mit Schlüssel k. Beim Start des Protokolls gibt es bereits die Schlüssel k_{AT} und k_{BT} zwischen Alice bzw. Bob und dem Server. Das Ziel des Protokolls ist ein gemeinsamer Schlüssel k_{AB} zwischen Alice und Bob. Alice beginnt das Protokoll, indem sie Bob eine Nachricht A, N_A schickt (Schritt 1), wobei "," Nachrichtenkonkatenation ist, A ein Bitstring, der Alice bezeichnet, und N_A ein Bitstring, der eine Zufallszahl bezeichnet. Bob versteht die Nachricht und bittet den Server um die Generierung eines Schlüssels (Schritt 2). Dabei ist T_B ein Zeitstempel in Form

eines Bitstrings. Der Server generiert den Schlüssel k_{AB} und schickt eine entsprechende Nachricht zu Alice (Schritt 3). Die kann den ersten Teil der Nachricht mit ihrem Schlüssel k_{AT} entschlüsseln. Den zweiten Teil kann sie nicht entschlüsseln, sondern leitet ihn an Bob weiter (Schritt 4). Sie authentifiziert sich dabei gegenüber Bob, indem sie die Zufallszahl N_B mit dem neuen gemeinsamen Schlüssel verschlüsselt.

Beim Verschicken von Nachrichten kann ein Empfänger typischerweise an der Nachricht selbst nicht erkennen, wer die Nachricht geschickt hat. Deshalb authentifiziert der Nachrichtenteil $E_{k_{\mathrm{AB}}}(N_B)$ Alice gegenüber Bob.

Aufgabe 2: Angriff

Es gibt verschiedene Ansätze, wie Angreifer ein Protokoll „brechen“, also in den Besitz eines geheimen Schlüssels kommen können. Nehmen wir für das obige Schlüsseletablierungsprotokoll an, dass sich die Schlüssel selbst (k_{AT}, k_{BT}, k_{AB}) nicht brechen lassen. Wie könnte eine Angreiferin Eve trotzdem in den Besitz des geheimen Schlüssels zwischen Alice und Bob kommen? Dabei soll das Protokoll nur genau einmal zwischen Alice und Bob ausgeführt werden, d.h., es wird nur ein gemeinsamer Schlüssel etabliert. Eve kann Nachrichten an beliebige Empfänger abfangen und selbst Nachrichten von beliebigen Absendern schicken. Alice und Bob können nicht unterscheiden, ob sie mit Eve oder dem jeweiligen Partner Nachrichten austauschen.

Aufgabe 3: Sichere bereits geschickte Nachrichten

Die Annahme, dass Schlüssel nicht gebrochen werden können, ist nicht realistisch. Es ist nur eine Frage der zur Verfügung stehenden Zeit. Wie könnte ein Protokoll aussehen, bei dem, falls ein Angreifer irgendwann einen Schlüssel bricht, er trotzdem nicht alle bis dahin geschickten Nachrichten lesen kann? Entwerfen Sie ein solches Protokoll, das beliebig viele Nachrichten von Alice zu Bob und eine Antwort übertragen soll, und versuchen Sie dabei, mit möglichst wenigen Nachrichten auszukommen. Dabei können Sie davon ausgehen, dass es ein kurzes Protokoll gibt, um einen neuen Schlüssel zwischen Bob und Alice zu etablieren:

$$\begin{aligned} &(1) \quad A \to B : k_A \\ &(2) \quad B \to A : k_B \end{aligned}$$

Das Protokoll funktioniert so, dass sich Alice und Bob jeweils einen (neuen) öffentlichen Schlüssel zusenden, mit dessen Hilfe und ihrem geheimen Schlüssel dann ein gemeinsamer geheimer Schlüssel $k_A \circ k_B$ berechnet werden kann.

Aufgabe 4: Sichere zukünftige Nachrichten

Auch die zeitlich entgegengesetzte Richtung zur vorherigen Aufgabe ist wünschenswert. Wie könnte ein Protokoll aussehen, bei dem, falls ein Angreifer irgendwann einen Schlüssel bricht, er trotzdem nicht alle zukünftigen Nachrichten lesen kann? Nehmen Sie an, dass es nach der Entschlüsselung noch mindestens eine nicht manipulierte Nachricht gab. Was würde passieren, wenn es diese eine Nachricht nicht gab?

Aufgabe 5: NP-vollständige Probleme

Man könnte auf die Idee kommen, NP-vollständige Probleme zur Verschlüsselung zu nutzen. Nehmen wir an, wir wollen aussagenlogische Erfüllbarkeit dafür nutzen. Diskutieren Sie diesen Ansatz.