Kryptographische Prüfsummen

Kryptografische Prüfsummen sind Algorithmen, die man auf beliebig lange Strings oder Dateiinhalte anwenden kann und die sehr schnell eine Zahl (= Prüfsumme, Hashwert, Message Digest) zurückliefern. Diese Verfahren müssen unbedingt folgende Eigenschaften haben:

1) kleinste Änderungen einer Datei -> große Änderungen der Prüfsumme („Unstetigkeit“)
2) "Nicht invertierbar": Zu einer gegebenen Prüfsumme darf sich keine Datei "schnell" finden lassen => Prüfsumme muss lang sein, sonst kann man das mit Brute Force Attacken schaffen.
3) Es soll in der Praxis nie vorkommen, dass 2 verschiedene Dateien dieselbe Prüfsumme haben. Natürlich ist es klar, dass es immer solche „Zwillinge“ geben muss, nur sollte ihr tatsächliches Auftreten extrem selten sein (d.h. kein solcher Fall sollte jemals dokumentiert sein).

Idee: PS(A) = PS(B) => A=B

128 Bit Prüfsummen reichen heute NICHT mehr aus (Geburtstagsphänomen s.u.!)

Verfahren:

- MD5 "Message Digest" (128 Bit, daher obsolet)
- SHA "Secure Hashing Algorithm" (160 Bit, gilt als obsolet)
- SHA-256 (256 Bit, das heutige Standardverfahren), SHA-384 (384 Bit-Version)
- SHA-3: ein neues Verfahren, das wie AES unlängst in einem öffentlichen Ausscheideverfahren ermittelt wurde und derzeit in alle Betriebssysteme integriert wird.

Geburtstagsphänomen: Bei Klassen mit 23 Schülern ist die Wahrscheinlichkeit größer als 50%, dass 2 Schüler am selben Tag des Jahres Geburtstag haben. Grund dafür ist, dass die Chancen quadratisch mit der Schülerzahl wachsen.

Umformulierung: Prüfsumme einer Datei <-> Geburtstag

Datei A ist das Original

Datei B ist die Fälschung; Hacker wollen: PS(B) = PS(A), d.h. 2 „Zwilling“

schlechter Algorithmus: man fixiert zuerst A und muss dann B berechnen:

n Versuche für B ergeben nur n Chancen auf einen Zwilling von A

besser:

A variieren zu A_1, A_2, ..., $A_{n/2}$ n/2 Versuche (z.B. Leerzeichen anhängen)

B variieren zu B_1, B_2, ..., $B_{n/2}$ n/2 Versuche
suche i, j mit: $PS(A_i) = PS(B_j)$ $n^2/4$ Chancen, Zahl steigt quadratisch mit n

Durch den quadratischen Angriff halbiert sich quasi die Sicherheit einer Prüfsumme bei Brute-Force-Attacken:

128 Bit symmetrischer Schlüssel	-> Sicherheit 128 Bit	-> sehr sicher
128 Bit Prüfsumme	-> Sicherheit 64 Bit	-> in Sekunden knackbar

Hashverfahren werden heute eingesetzt, um die Identität zweier Dateien A, B zu beweisen, die nicht am selben Ort gespeichert sind und somit nicht direkt verglichen werden können. Man berechnet von beiden Dateien den Hashwert und vergleicht diese.

Signatur

Eine Signatur ist eine kryptografische Prüfsumme eines Dokuments, asymmetrisch verschlüsselt mit dem **Private Key** des Unterzeichners. Deshalb ist nur dieser in der Lage, eine persönliche Signatur zu erzeugen. Weil zur Überprüfung der Signatur das Schlüsselpaar gültig sein muss (auch noch nach Jahren), verwendet man hauptsächlich RSA für Signaturen. Die Signatur kann in einer Extradatei getrennt von den signierten Daten oder auch in einem Paket mit den Daten enthalten sein.

Signatur eines Dokuments erstellen:

1) Berechnung eines Hashwerts des Dokuments mit einem geeigneten Verfahren (z.B. SHA-256).
2) RSA-Verschlüsselung der Prüfsumme mit dem **Privaten** Schlüssel.

Signatur eines Dokuments (Dokument, Signatur und Signatur-Algorithmus liegen vor) verifizieren:

1) **Public** Key des Signierers besorgen (RSA).
2) Signatur damit entschlüsseln -> man erhält die Prüfsumme der Datei.
3) Prüfsumme für das Dokument mit dem richtigen Hashverfahren neu berechnen, diese muss mit jener aus Schritt 2) übereinstimmen.

Test beweist:

1) Das Dokument wurde seit dem Signieren nicht verändert (wegen Hashverfahren).
2) Der Hersteller der Signatur besitzt den richtigen Private Key, ist also in den allermeisten Fällen (siehe NSA) die richtige Quelle.

Heute werden fast alle Software-Produkte durch Signaturen gesichert. Dadurch ist sichergestellt, dass der Kunde immer die Orginalsoftware des Herstellers erhält, wie auch immer er in den Besitz dieser Software gekommen ist.