Integrität und Authentizität

Werden nicht durch Verschlüsselung bereitgestellt!

- Integrität Problem bei Verschlüsselung von nicht redundanten Daten (komprimierte Dateien, geheime Schlüssel).

Idee: Geeignete kryptographische Prüfsummen mitschicken.

Davon gibt es zwei Typen:

- Manipulation Detection Codes (MDC), Hashfunktionen ohne Schlüssel. Liefert Fingerprint oder Message Digest.
- Message Authentication Codes (MAC), Hashfunktionen mit Schlüssel.

Hashfunktionen haben vielfache Verwendung (bei Verschlüsselung, Unterschriften, Pseudozufallszahlen, Wörterbüchern ...)

Sicherheitsmerkmale von Hash Fnktn

Seien X,Y Mengen und $f : X \to Y$. Bildwerte $f(x)$ sollen mit einem Algorithmus effizient berechenbar sein.

Aufgaben:

1. (Urbild berechnen) Zu $y \in Y$ ein $x \in X$ mit $f(x) = y$ bestimmen.
2. (Zweites Urbild berechnen) Zu $x \in X$ ein $x' \in X$ mit $x' \neq x$ und $f(x) = f(x')$ bestimmen.
3. (Kollision finden) $x, x' \in X$ mit $x \neq x'$ und $f(x) = f(x')$ bestimmen.

Def: Eine Funktion heißt Einwegfunktion, wenn es keinen effizienten Algorithmus gibt, der Aufgabe 1 mit signifikanter Wahrscheinlichkeit löst.

Def: Eine Funktion heißt schwach kollisionsresistent (kollisionsresistent), wenn es keinen effizienten Algorithmus gibt, der Aufgabe 2 (Aufgabe 3) mit signifikanter Wahrscheinlichkeit löst.

Effizient und signifikant

Effiziente Laufzeit und signifikante Erfolgswahrscheinlichkeit eines Algorithmus A kann man quantitativ oder qualitativ angeben.

Quantitativ: Für eine speziell vorgelegte Situation, z.B. effizient = Laufzeit $\leq 2^{40}$ Bitops, signifikant = Erfolgswahrscheinlichkeit $\geq 2^{-40}$ ...

Qualitativ: Gemessen in der Bitlänge k der Eingabe von A.
Dann effizient = Laufzeit polynomiell in k, signifikant = Erfolgswahrscheinlichkeit nicht vernachlässigbar in k.

Wenn man in der Kryptographie sagt, daß es keinen in k effizienten Angreifer A mit signifikanter Erfolgswahrscheinlichkeit gibt, dann nennt man k häufig Sicherheitsparameter.

Polynomiell und vernachlässigbar

Eine Funktion $g : \mathbb{R}^{\geq 0} \to \mathbb{R}$ heißt polynomiell (in k), wenn es ein Polynom $P \in \mathbb{R}^{\geq 0}[x]$ und $k_0 \in \mathbb{R}^{\geq 0}$ gibt, so daß $|g(k)| \leq P(k)$ für alle $k \geq k_0$.

Eine Funktion $g : \mathbb{R}^{\geq 0} \to \mathbb{R}$ heißt vernachlässigbar (in k), wenn es für jedes Polynom $Q \in \mathbb{R}^{\geq 0}[x]$ ein $k_0 \in \mathbb{R}^{\geq 0}$ gibt, so daß $|g(k)| \leq 1/Q(k)$ für alle $k \geq k_0$.

Zurück zu Einweg- und kollisionsresisten Funktionen

Um von Einweg- und kollisionsresistenten Funktionen im qualitativen Sinn sprechen zu können, betrachtet man eine Familie von Funktionen $f_k : X_k \to Y_k$ und $k \to \infty$.

Es soll einen Algorithmus A geben, welcher Bildwerte $f_k(x)$ nach Eingabe von k, x effizient berechnet.

Auf der anderen Seite soll es keinen Algorithmus B geben, der Urbilder oder Kollisionen mit signifikanter Wahrscheinlichkeit unter Eingabe von k für signifikant viele y bzw. x, x' effizient berechnen kann.

Hieraus folgen beispielsweise Größenbedingungen an X_k und Y_k in Abhängigkeit von k.

Hash- und Kompressionsfunktionen

Def: Seien X, Y, Z Mengen mit $\#Z < \#Y < \infty$ und $\#X = \infty$.
Eine Hashfunktion ist eine Funktion $h : X \to Z$.
Eine Kompressionsfunktion ist eine Funktion $h : Y \to Z$.

Meistens $X = \{0,1\}^*$, $Y = \{0,1\}^m$, $Z = \{0,1\}^n$ mit $m > n$.
Sind nicht injektiv!

Wieder Forderung:
Bildwerte von Hash- und Kompressionsfunktionen sollen effizient durch Algorithmen nach Eingabe der Argumente berechnet werden können.

Im folgenden verstehen wir unter Hashfunktionen auch Kompressionsfunktionen.

Zwei Anwendungen

1. Paßworte:
   - In der Paßwortdatei sind nur die Hashwerte der Paßwörter gespeichert.
   - Beim Einloggen wird der Hashwert des eingegeben Paßworts berechnet und mit dem gespeicherten verglichen.
   - Paßwortdatei soll nicht unbedingt lesegeschützt sein $\Rightarrow$ Hashfunktion soll Einwegfunktion sein.
   - Aber: Dictionary Angriffe (mögliche Paßworte ausprobieren und Hashwerte vergleichen).
2. Unterschriften:
   - Nicht das Dokument, sondern nur den Hashwert unterschreiben.
   - Angreifer soll kein zweites Dokument mit dem gleichen Hashwert berechnen können $\Rightarrow$ Hashfunktion soll kollisionsresistent sein.

Zufallsorakelmodell

Auch Random Oracle Model (RO). Ist theoretische Herangehensweise.

Hashfunktionen werden idealisiert als zufällige Funktionen modelliert.
- Ermöglicht und vereinfacht Untersuchungen und Sicherheitsbeweise kryptographischer Verfahren, welche Hashfunktionen verwenden.

Für in der Praxis verwendete Hashfunktionen kann man die Einwegeigenschaft oder die Kollisionsresistenz nicht beweisen.
- Hashfunktionen im RO haben diese Eigenschaften.

Man kann eine zufällige (Hash)Funktion nicht effizient als ganzes beschreiben.
- Daher Realisierung/Simulierung durch ein Orakel.

Hashsimulation durch Orakel

Bei einer Anfrage nach dem Hashwert von x geht das Orakel wie folgt vor: Das Orakel überpüft, ob $h(x)$ schon einmal erfragt und berechnet wurde, wenn

- ja, dann wird dieser Wert zurückgegeben.
- nein, dann wird ein zufälliger Wert zurückgegeben und als $h(x)$ gespeichert.

Insofern wird wirklich eine zufällige Funktion $h : X \to Z$ definiert.

Eine Orakelanfrage zählt in der Laufzeit eines Algorithmus als ein Schritt (konstante Zeit). Die Anzahl der Orakelanfragen wird üblicherweise angegeben und sollte polynomiell sein.

Angriffe im Zufallsorakelmodell

Sei $h : X \to Z$ mit $\#Z = n$ Hashfunktion.

Angriff auf Einweg-Eigenschaft von h:

- Angreifer berechnet s Hashwerte durch Anfragen an das Orakel.
- Mit Wahrscheinlichkeit $(1-1/n)^s$ ist Zielwert y nicht darunter.
- Mit Wahrscheinlichkeit $(1-(1-1/n)^s)+(1-1/n)^s/n$ kann er das richtige Urbild x raten.

Angriff auf die schwache Kollisonsresistenz von h:

- Ähnlich wie bei der Einweg-Eigenschaft.

Aufgrund des Zufallsorakelmodells gibt es auch keine Strategien, die eine bessere Erfolgswahrscheinlichkeit haben würden.

Angriffe im Zufallsorakelmodell

Schreibe $n = 2^k$.

- Der Beitrag $(1-1/n)^s/n$ ist vernachlässigbar in k, unabhängig von s.
- Für einen in k polynomiellen Algorithmus muß s polynomiell sein. Wegen $(1-1/n)^s \geq 1-s/n$ ist $1-(1-1/n)^s \leq s/n$. Dies ist ebenfalls vernachlässigbar in k.
- Damit ist die Erfolgswahrscheinlichkeit vernachlässigbar, wenn s nur polynomiell ist.
- Umgekehrt sind ungefähr 2^k Orakelanfragen erforderlich, um eine konstante Erfolgswahrscheinlichkeit zu haben.

Folgerung: Eine Hashfunktion im Zufallsorakelmodell ist eine Einwegfunktion und schwach kollisionsresistent.

Geburtstagsparadoxon

Thm: Wählen wir aus $n \geq 2^{16}$ Elementen $k \geq 1.18\sqrt{n}$ zufällig mit Zurücklegen aus, so haben wir mit Wahrscheinlichkeit $> 1/2$ mindestens ein Element mindestens zweimal ausgewählt.

Bew: Sei X_i das Ereignis, daß das i-te gewählte Element nicht mit einem der vorherigen übereinstimmt. Dann gilt $\Pr(X_i \mid X_1,\ldots,X_{i-1}) = 1-(i-1)/n$ und $\Pr(X_1,\ldots,X_k) = \prod_{i=1}^{k}(1-(i-1)/n)$. Aus $1+x \leq e^x$ folgt $\prod_{i=1}^{k}(1-(i-1)/n) \leq \prod_{i=1}^{k} e^{-(i-1)/n} \leq e^{-k(k-1)/(2n)}$. Für das angegebene n und k gilt $e^{-k(k-1)/(2n)} < 1/2$. Somit erhalten wir eine Doppelauswahl mit Wahrscheinlichkeit $> 1/2$. $\square$

Auf der anderen Seite ergibt sich aus $\prod_{i=1}^{k}(1-(i-1)/n)$ $\geq (1-k/n)^k \geq 1-k^2/n$ die obere Schranke k^2/n für die Wahrscheinlichkeit einer Doppelauswahl.

Angriffe im Zufallsorakelmodell

Angriff auf die Kollisionsresistenz von h:

- Angreifer berechnet $s \geq 1.18\sqrt{n}$ Hashwerte durch Anfragen an das Orakel.
- Befindet sich unter den Hashwerten eine Kollision, so wird diese ausgegeben.

Nach dem Geburtstagsparadoxon ist die Wahrscheinlichkeit hierfür $> 1/2$. Speicherbedarf ca. $\sqrt{n}$.

Zum Speichern der Hashwerte eine Tabelle anlegen und nach den ersten $\log_2(\sqrt{n})$ Bits der Hashwerte indizieren. Beim Suchen direkt im entsprechenden Tabellenfeld nachschauen, ob bereits Hashwert definiert bzw. das zugehörige x eingetragen wurde.

$\Rightarrow$ Hash Sort

Floyd’s Algorithmus

Speicherbedarf sehr groß. Können auch noch besser vorgehen: Floyd’s Algorithmus zum Finden von Zykeln.

$h : X \to Z$ Hashfunktion.
Annahme $Z \subseteq X$, $x_0 \in X \setminus Z$.

Definiere $x_{i+1} = h(x_i)$.

- Es gibt minimale $\alpha < \beta$ mit $x_\alpha = x_\beta$ und $x_{\alpha-1} \neq x_{\beta-1}$. Dann $x_{\alpha+i} \neq x_{\beta+i}$ für alle $i < 0$ und $x_{\alpha+i} = x_{\beta+i}$ für alle $i \geq 0$.
- Daher Zykellänge $\delta = \beta - \alpha$.
- Für $i \geq 1$ gilt $x_i = x_{2i} \Leftrightarrow i \geq \alpha$ und $2i \equiv i \bmod \delta$ bzw. $i \equiv 0 \bmod \delta$.

$\Rightarrow$ Das minimale i mit $x_i = x_{2i}$ erfüllt $i \leq \alpha + \delta - 1$.
$\Rightarrow$ Das minimale i mit $x_i = x_{i+r\delta}$ für $r \geq 1$ erfüllt $i = \alpha$.

Floyd’s Algorithmus

Daraus ergibt sich folgendes Vorgehen:

- Berechne $(x_1, x_2), (x_2, x_4), \ldots$ bis $x_i = x_{2i}$. Setze $\delta' \leftarrow i$.
- Berechne $(x_1, x_{1+\delta'}), (x_2, x_{2+\delta'}), \ldots$ bis $x_i = x_{i+\delta'}$. Setze $\alpha' \leftarrow i$.
- Dann $x_{\alpha'-1} \neq x_{\alpha'-1+\delta'}$ und $x_{\alpha'} = x_{\alpha'+\delta'}$, also $h(x_{\alpha'-1}) = h(x_{\alpha'-1+\delta'})$.

Sei $n = \#Z$.

Im Zufallsorakelmodell erfolgt eine Kollision nach $1.18\sqrt{n}$ vielen Hashwerten x_i mit Wahrscheinlichkeit $\geq 1/2$. In diesem Fall daher $\alpha + \delta \leq 1.18\sqrt{n}$.

Anzahl der Orakelaufrufe: $3\delta' + 2\alpha' \leq 3(\alpha + \delta - 1) + 2\alpha = O(\sqrt{n})$.
Speicheraufwand: $O(1)$.

Folgerung

Sei $h : X \to Z$ eine Hashfunktion mit $\#Z = n = 2^k$.

Im Zufallsorakelmodell kann man also eine Kollision nach ca. $s = \sqrt{n}$ Anfragen an das Hashorakel mit guter Wahrscheinlichkeit finden.

Auf der anderen Seite ist diese Wahrscheinlichkeit durch s^2/n nach oben beschränkt. Benutzt man weniger Orakelanfragen, nimmt die Wahrscheinlichkeit einer Kollision zügig (quadratisch) ab.

Ein in k polynomieller Angreifer kann folglich auf eine Kollision nur mit vernachlässigbarer Wahrscheinlichkeit $\mathrm{poly}(k)/2^k$ hoffen. Seine Erfolgswahrscheinlichkeit ist damit insgesamt ebenfalls vernachlässigbar.

Folgerung

Damit sind Hashfunktionen im Zufallsorakelmodell auch kollisionsresistent, wenn gleich der Aufwand zum Finden einer Kollision wesentlich geringer ist als der, Urbilder zu berechnen.

Im Zufallsorakelmodell ergibt sich zusammenfassend:

- k Bit Sicherheit bezüglich der Einweg-Eigenschaft.
- Nur $k/2$ Bit Sicherheit bezüglich Kollisionsresistenz.

Von einer „guten" Hashfunktion fordert man daher im Standardmodell (d.h. nicht im Zufallsorakelmodell), daß Urbilder und Kollisionen nur mit Aufwand ungefähr 2^k bzw. $2^{k/2}$ berechnet werden können sollen.

In der Praxis fordert man zur Zeit $k \geq 160$.