Themen zur Computersicherheit

Teilnehmerauthentisierung

PD Dr. Reinhard Bündgen
buendgen@de.ibm.com

Ziel der Authentisierung

- Ein Teilnehmer der Zugang zu einem System anfordert muss seine Identität nachweisen
- Teilnehmer wird im System durch Teilnehmer-Identifikator (Mitgliedsnummer, Kontonummer, e-mail Adresse o.ä.) beschrieben
  - evtl mit Attributen (wie Adresse, Geburtsdatum, Kreditkartennummer,…) versehen
- Die Verifikation, dass die Teilnehmerattribute wirklich zum Teilnehmer gehören
  - außerhalb des Systems, z.B. persönliche Anmeldung, Postident-Verfahren
  - Plausibilitätschecks: Rückfrage an e-mail Adresse, Konsistenz von Kreditkartendaten
  - elektronische Signaturen (ePA)

Richtung der Authentisierung

- Teilnehmer beweist dem System seine Identität
  - z.B login
- System beweist dem Teilnehmer seine Identität
  - z.B. die meisten HTTPS Server
- einseitige Authentisierung
  - z.B klassisches login
- beidseitige Authentisierung
  - z.B.ssh

Methoden der Authentisierung

- Besitz
  - z.B. Ausweis, Smartcard, Handy
  - → OTP
- Wissen
  - Passwort, PIN, TAN-Liste
- (körperliche) Merkmale
  - Irismuster, Fingerabdruck, Retina, Handschrift, Stimme, Handgeometrie, DNA?,
- Lokation
  - GPS Daten, Konsole, Netzwerksegment
- Einfaktor-Authentisierung
- Mehrfaktor-Authentisierung
- Vier-Augenprinzip

Passwortverfahren - Anforderungen

- Passwort: auch Kennwort, Passphrase, PIN
- Anforderungen
  - nur Teilnehmer darf Passwort kennen
  - System muss Passwort erkennen können
  - System muss unterschiedliche Passwörter unterscheiden können
- abgeleitete Anforderungen
  - Passwort darf nur verschlüsselt gespeichert werden
  - Passwort darf nur verschlüsselt übertragen werden

Kryptographische Hashfunktionen

im folgenden gilt $\sum = \{0,1\}$

Definition Eine Funktion $h: \sum^* \rightarrow \sum^n$ heißt *Hashfunktion* wenn es einen effizienten Algorithmus gibt, für jedes $m \in \sum^*$ den Wert $h(m)$ berechnet.

Definition Eine Hashfunktion h heißt *Einweg-Hashfunktion*, wenn zu zufällig gewähltem $y \in \sum^n$ der Wert $h^{-1}(y)$ nicht effizient berechnet werden kann

Definition Eine Hashfunktion h heißt *schwach kollisionsresistent (target collision resistent, 2nd pre-image resistence)* wenn es keinen effizienten Algorithmus gibt der zu gegebenen $m \in \sum^*$ ein $m' \in \sum^*$ findet mit $m \neq m'$ und $h(m) = h(m')$.

Definition Eine Hashfunktion h heißt *stark kollisionsresistent* wenn es keinen effizienten Algorithmus gibt der $m \in \sum^*$ und $m' \in \sum^*$ findet mit $m \neq m'$ und $h(m) = h(m')$.

Definition Eine stark kollisionsresistente Einweg-Hashfunktion heißt *kryptographische Hashfunktion*.

Ideale kryptographische Hashfunktionen

- Charakterisierung (Ferguson et al): Eine ideale Hashfunktion verhält sich wie eine Zufallsabbildung.
- Eine Attacke auf eine Hashfunktion: eine nicht generische Methode, die einen Unterschied zu einer idealen Hashfunktion findet.

Das Geburtstagsparadox

- **Satz** Zu beliebiger Hashfunktion h: $\sum^* \rightarrow \sum^n$ gibt es einen Algorithmus der zur Berechnung von $h^{-1}(y)$ für ein zufällig gewähltes $y \in \sum^n$ im Durchschnitt 2^{n-1} Hashoperationen benötigt
- **Satz** (*Geburtstagsparadox*) Zu beliebiger Hashfunktion h: $\sum^* \rightarrow \sum^n$ ist die Wahrscheinlichkeit P(n,k), dass eine Menge $M \subseteq \sum^*$ mit $|M| = k \geq 1{,}2 \cdot 2^{n/2}$ zwei Elemente m, m' $\in \sum^*$ mit $m \neq m'$ und $h(m) = h(m')$ enthält, größer als 0,5.
- Beweis:

$$1-P(n,k)=\prod_{i=0}^{k-1}\frac{2^n-i}{2^n}\leq\prod_{i=0}^{k-1}e^{\frac{-i}{2^n}}\leq e^{\frac{-(k-1)^2}{2\cdot 2^n}}\leq e^{\frac{-(k-1)^2}{2\cdot 2^n}}$$

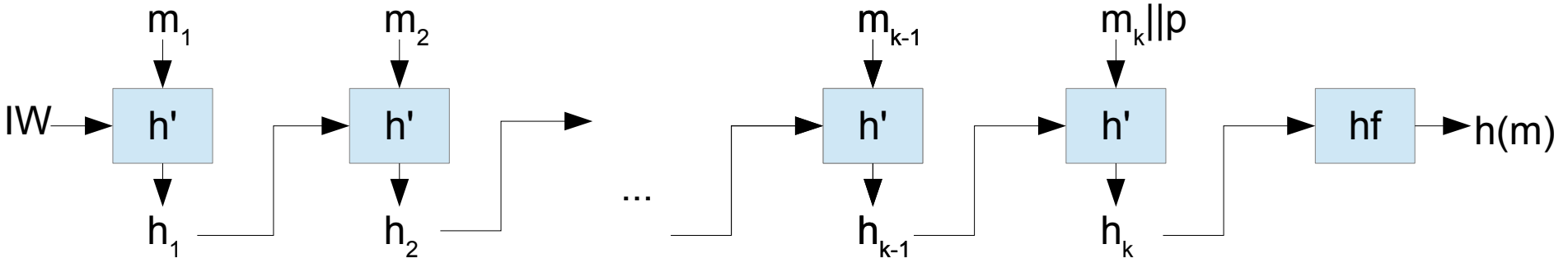
$$1-P(n,1{,}2\cdot 2^{n/2}+1)\leq e^{-\frac{2\cdot 2^n\cdot\ln 2}{2\cdot 2^n}}=e^{-\ln 2}=1/2$$

- **Geburtstagsangriff**:
  - Berechne die Hashwerte von $1{,}2 \cdot 2^{n/2}$ zufällig gewählten Elementen aus $\sum^*$.

Meet-in-the-middle Angriff

- Sei die Hashfunktion h auf dem eingeschränkten Definitionsbereich $\sum^n$ bijektiv
- $D \subseteq \sum^n$ mit $|D| = 2^d$
- $h|_D : D \rightarrow \sum^n$ eine vorberechnete Tabelle von Hashwerten
- dann ist im Durchschnitt jeder 2^{n-d}-te Hashwert in D enthalten
- Angriff
  - wähle D mit $|D| = 2^{n/2}$
  - Berechne $h|_D(m)$ für alle $m \in D$: $2^{n/2}$ Operationen
  - sortiere h(D): $n/2 \cdot 2^{n/2}$ Operationen
  - wähle zufällige Menge $E \subseteq \sum^n \setminus D$ mit $|E| = 2^{n/2}$
  - Berechne h(m) für alle $m \in E$: $2^{n/2}$ Operationen
  - suche h(m) in h(D): je n/2 Operationen
  - Kollision wird mit Wahrscheinlichkeit ≥ 0,5 gefunden
  - Aufwand: $n \cdot 2^{n/2}$

Typische Struktur von Hashfunktionen



- Hashfunktion h:$\sum^* \rightarrow \sum^n$ wird iterativ mit elementarer Hashfunkton h': $\sum^k$ x $\sum^b \rightarrow \sum^k$ berechnet mit k >= n
- hf: $\sum^k \rightarrow \sum^n$ ist eine finale Filterfunktion
- Teile Nachricht m in Blöcke der Länge b auf:
  - $m = m_1 \,||\, m_2 \,||\, \ldots \,||\, m_{k-1} \,||\, m_k$ mit $|m_i| = b$ für $i<k$ und $|m_k| \leq b$
- sei $p \in \sum^*$ mit $|p| = b-|m_k|$ ein Padding
- sei IW $\in \sum^k$ ein fester Initialisierungswert
- $h_0 = IW$
- $h_i = h'(h_{i-1}, m_i)$ für $1 \leq i \leq k-1$
- $h_k = h'(h_{k-1}, m_k||p)$
- $h(m) = hf(h_k)$

Notation:
|| ist der Konkatenationsoperator

Einige wichtige Hashfunktionen

Name	Autor	Jahr	Block größe	Hashgrößen (bit)	Aufwand Kollision
MD4	Rivest	1990	521	128	1995:2^{20} (heute 2?)
MD5	Rivest	1992	512	128	2009: 2^{16}
SHA-1	NSA	1995	512	160	2007: 2^{61}
SHA-2	NIST	2002	512, 1024	224,256, 384,512 [1)]	
SHA-3 (Keccak)	Bertoni et al	2011		224,256, 384,512	

Empfohlene sichere Hashlängen (www.keylenght.com)
je nach Autor/Organisation

- für 2014: 155 – 224 bits (BSI 224 bits)
- für 2020: 163 – 256 bits (BSI 256 bits)
- für 2030: 186 – 256 bits
- für 2050: 203 – 512 bits

[1)] Hashgrößen passen zu 2DES, AES-128, AES-192/3DES, AES-256 Chiffren

Passwortverfahren - Methoden

- System hat Passwortdatenbank pdb: T → EP die jedem autorisierten Teilnehmer t ∈ T ein verschlüsseltes Passwort in EP zuweist.
- Passwort $p_t \in P$ des Teilnehmers t wird mit Passwortverschlüsselungsfunktion ep: P → EP verschlüsselt und mit hinterlegten verschlüsselten verglichen: $ep(p_t) = pdb(t)$?
- Folgerung
  - ep^{-1}:EP → P mit $ep^{-1}(ep(p)) = p$ muss *praktisch nicht berechenbar* (also sehr aufwendig) sein → *Einwegfunktion*
  - ep muss jedoch mit vertretbarem Aufwand berechenbar sein

Passworthasfunktionen

- bekannte Funktion gemäß Kerkhoffs Prinzip
- Passwortverschlüsselung: kryptographische Hashfunktion
- dasselbe Passwort muss auf unterschiedlichen Systemen unterschiedliche Hashes ergeben: Salz
- sie muss schwer zu knacken sein: einstellbarer Kostenfaktor mehrfaches Iterieren einer zugrundeliegenden Hashfunktion
- z.B. PBKDF2 aus PKCS#5 v2 oder bcrypt

PBKDF2

- Password based key derivation function 2
- PBKDF2: $\sum^* \times \sum^* \times \mathbf{N} \times \mathbf{N} \rightarrow \sum^*$ berechnet Passworthash (bzw den Schlüssel)
- F: $\sum^* \times \sum^* \times \mathbf{N} \times \mathbf{N} \rightarrow \sum^n$ berechnet Schlüsselteil
- PRF: $\sum^* \times \sum^* \rightarrow \sum^n$ berechnet ein Hashwert einer Nachricht, der von einem Schlüssel abhängt
- PBKDF2(pw,salt,iter,keylen) = $T_1||...||T_{m-1}||msb(T_m,r)$
- mit
  - T_j = F(pw,salt,iter,j)
  - $m=\lceil keylen/n \rceil$; r = keylen – (m—1)•n
  - $F(p,s,it,j) = U_1 \oplus ... \oplus U_{it}$ mit
    - U_1=PRF(pw, salt || INT(j))
    - U_i=PRF(pw,U_{i-1})

Notation:
⊕ ist der bitweise xor Operator

Wie werden Passwörter geknackt I

- online Angriff
  - Passwort raten + Versuch einzuloggen
- offline Angriff
  - Gegeben ein verschlüsseltes Passwort suche Urbild zu bekannter Passworthashfunktion
  - Gegeben eine Liste verschlüsselter Passwörter suche Urbild zu einem der verschlüsselten Passwörter

Gegenmaßnahmen

- online Angriff:
  - erlaube nur endliche Zahl von Fehlversuchen
  - forciere nach einem Fehlversuch eine Pause, Pausen werden mit jedem Fehlversuch größer
  - jailing: vorspielen eines gelungenen Einbruchs
  - CAPTCHAs (Completely Automated Public Turing Test to tell Computers and Humans Apart): verhindern automatischer Angriffe
- offline Angriff
  - Passworthashes (/etc/shadow) dürfen nur autorisierten Lesern zugänglich sein
  - langsame Passworthashfunktionen
  - (systemspezifisch) salzen

Wie werden Passwörter geknackt II

- Katalog von vorberechneten Hash-Urbildern (rainbow tables)
- Passwortkataloge
  - beliebte Passwörter (z.B. 12345678, letmein, asdfghj)
  - Wörterbücher verschiedener Sprachen
  - aussprechbare Buchstabenketten
  - jemals geknackte Passwörter
- Variationen von Katalogelementen
  - Prefixe, Suffixe,
  - Ersetzung Buchstaben durch Ziffern, Sonderzeichen (z.B. i → 1, s → $, at → @, for → 4)
  - Transformationen: vorwärts, rückwärts
- Indexierung von Festplatten
- Nutzung persönlicher Informationen
  - Phishing

Beispiele gefundener Passwörter

ilovemySister31

windermere2313

k1araj0hns0n

Qbesancon321

gonefishing1125

Apr!l221973

ilovetofunot

allineedislove

Sh1a-labe0uf

tmdmmj17

all of the lights

iloveyousomuch

qeadzcwrsfxv1331

@Yourmom69

DG091101%

BandGeek2014

Philippians4:13

Philippians4:6-7

i hate hackers

Schneier newsletter 03/2014: „Last year, Ars Technica gave three experts a 16,000-entry encrypted password file, and asked them to break as many possible. The winner got 90% of them, the loser 62% -- in a few hours.“

Gute Passwörter

- Großer Zeichensatz: Buchtaben (groß und klein), Zahlen Sonderzeichen
- echte Zufalls Wörter ($36^8 > 10^{12}$)
  - Kontospezifisch verziert
- Anfangsbuchstaben von *persönlichen* Phrasen
  - Schneier: "This little piggy went to market" -> "tlpWENT2m"
  - keine Shakepearezitiate!
- Passwortwiederherstellungsfragen:
  - Antworten dürfen nicht unsicherer sein als Passwort → lügen!