Seminar Hacking: Passwortsicherheit / Hashes

Tobias Hardes / Marcel Kreuer

4. Dezember 2013

Inhaltsverzeichnis

Passwortklau

- November 2013 - Vudu (Video on demand)
  - Benutzerinformationen (Name, Adresse,...)
  - Verschlüsselte Passwörter
  - Nach Vudu schwer zu knacken
  - Befürchtet: Spam
- November 2013 - Adobe
  - 153 Millionen Zugangsdaten
  - Schlecht verschlüsselt
  - Schnell zu erratende Sicherheitshinweise
  - Passwort: “adobeadobe“ Hinweis: "company name twice"
- November 2013: Mobilfunkanbieter Simyo
  - Mitarbeiterzugriff auf ersten 4 Zeichen des Passworts
- November 2013: Code-Hoster GitHub
  - Passwortangriff mit 40.000 IP-Adressen
  - Sollte Passwörter der User knacken

Definition

Eine **Hashfunktion** ist eine Funktion h der Form

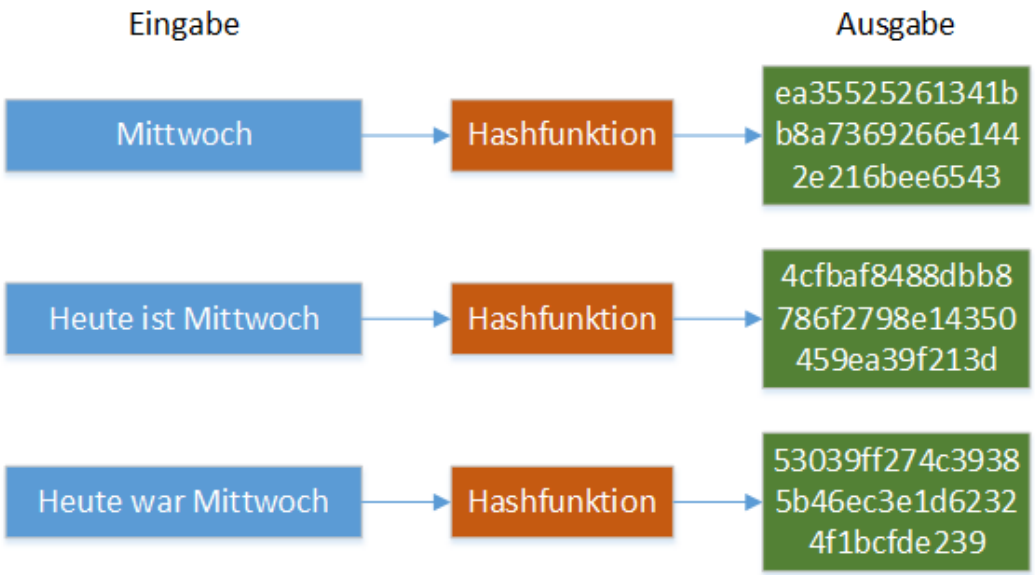
$$h : \Sigma^* \rightarrow \Sigma^n$$

wobei Σ eine endliche Menge und $n \in \mathbb{N}$.

h heißt **Einwegfunktion**, wenn kein Angreifer für ein zufälliges $y \in \Sigma^n$ mit hoher Wahrscheinlichkeit und mit vertretbaren Aufwand ein $x \in \Sigma^*$ zu finden, sodass h(x) = y.

Bemerkung:
Einwegfunktionen sind die Grundlage symmetrischer Kryptographie.

Hinweise



- **Praktisch unmöglich bzw. nicht effizient:**
  Finden des Urbilds benötigen zu viel Zeit oder Platz.
- Bei gegebenen x muss h(x) noch effizient berechenbar sein.
- h kann nicht injektiv, sondern nur surjektiv sein.

Sicherheit von Hashfunktionen

- Definition: Kollision
  - Paar (x,x') mit $x \neq x'$, aber h(x) = h(x')
  - Beispiel: $h(b_1, b_2, ..., b_n) = b_1 \oplus b_2 \oplus .. \oplus b_n$
  - Kollision: $h(0011) = h(1100)$
- Schwache Kollisionsresistenz („Zweites-Urbild-Resistenz“)
  - Für gegebenes x kann keine Kollision (x,x') mit h(x) = h(x') effizient berechnet werden.
  - z.B. Überprüfung von Daten mit einem Original
- Starke Kollisionsresistenz
  - Es ist nicht möglich eine Kollision h(x) = h(x') mit $x \neq x'$ zu finden
  - z.B. Digitale Unterschriften

Beispiel

- Alice führt Festplattensicherung durch.
- Alice möchte diese vor unbemerkten Veränderungen schützen (Integrität).
- Alice berechnet den Hashwert der Sicherung.
- Alice speichert den Hashwert auf einer sicheren Smartcard.
- Eve kann die Sicherung nicht verändert ohne, dass sich der Hashwert auch verändert
- Alice würde eine Änderung durch erneute Hashberechnung bemerken.

Authentifizierung - Prinzipien

- Problemstellung
  - Verifizierung einer Identität
- Verifizierung möglich durch:
  - Besitz
  - Biometrie
  - **Wissen**
  - Kombinationen

Authentifizierung - Wissen

- Geteiltes Geheimnis
  - Passwörter
  - Pin
  - Antworten auf Fragen
- Challenge-Response
  - Frage / Aufgabe $\longleftrightarrow$ Antwort
  - ChipTan, iTan,...
- Nachteile
  - Vergessen
  - Verraten
  - Erraten

Passwortarten

- Gewöhnliche Passwörter
  - Shared-Secrect → Beide Parteien teilen ein Geheimnis
  - Wird wiederverwendet → E-Mail, Login,...
  - Eher geringe Sicherheit
- One-Time-Passwörter
  - Jeder Zugang benötigt ein neues Passwort ⇒ RSA SecurID
  - Abgreifen (Man-in-the-middle) eines Passwortes gewährt nur ein Mal Zugang
  - Mehr Aufwand → Generierung, Verwaltung, Speichern von zusätzlichen Werten...

Abbildung: Quelle: www.tokenguard.com

Passwörter - Speicherung

- Sichere Speicherung notwendig!
- Beschränkter Zugang zum Passwortspeicher
- Praktische Lösung: Speicherung als Hashwert
- Beispiel Linux: $\rightarrow$ /etc/shadow

Passwörter - Angriffe

- Social Engineering
  - Fragen
  - Keylogger
  - Shouldersurfing
  - Gewaltandrohung
  - ...
- Angriffe via „Software“
  - Brute Force
  - Lookup-Table
  - Hellmann Tabelle
  - **Rainbowtable**

Rainbowtables (I) - Der Anfang

- Speicher alle Hashes mit Passwort in einer Tabelle
- Beispiel:
  - 64 Mögliche Zeichen für ein Passwort $[A - Za - z0 - 9./]$
  - Passwortlänge: 6 Zeichen
  - 64^6 potentielle Variationen
  - Speichern von Passwort und Hash $\Rightarrow$ 16 Byte für Hash und 6 Byte für Klartext
  - Speicherbedarf: ca. 1,4 TB Speicher(!) $\rightarrow$ Festplatte als Flaschenhals

Rainbowtables (II)

- Zwischenlösung: Speicher $\Leftrightarrow$ Laufzeit (time-memory tradeoff)
- Erstellen einer Kette
  - Hashfunktion - Hashen das Passworts
  - Reduktionsfunktionen - Hashwert ein potentielles Passwort zuordnen
  - $\Rightarrow$ Grund für den Namen Rainbowtable
- Gespeichert werden erster und letzter Wert der Kette. Der Rest wird berechnet.

Rainbowtables (III)

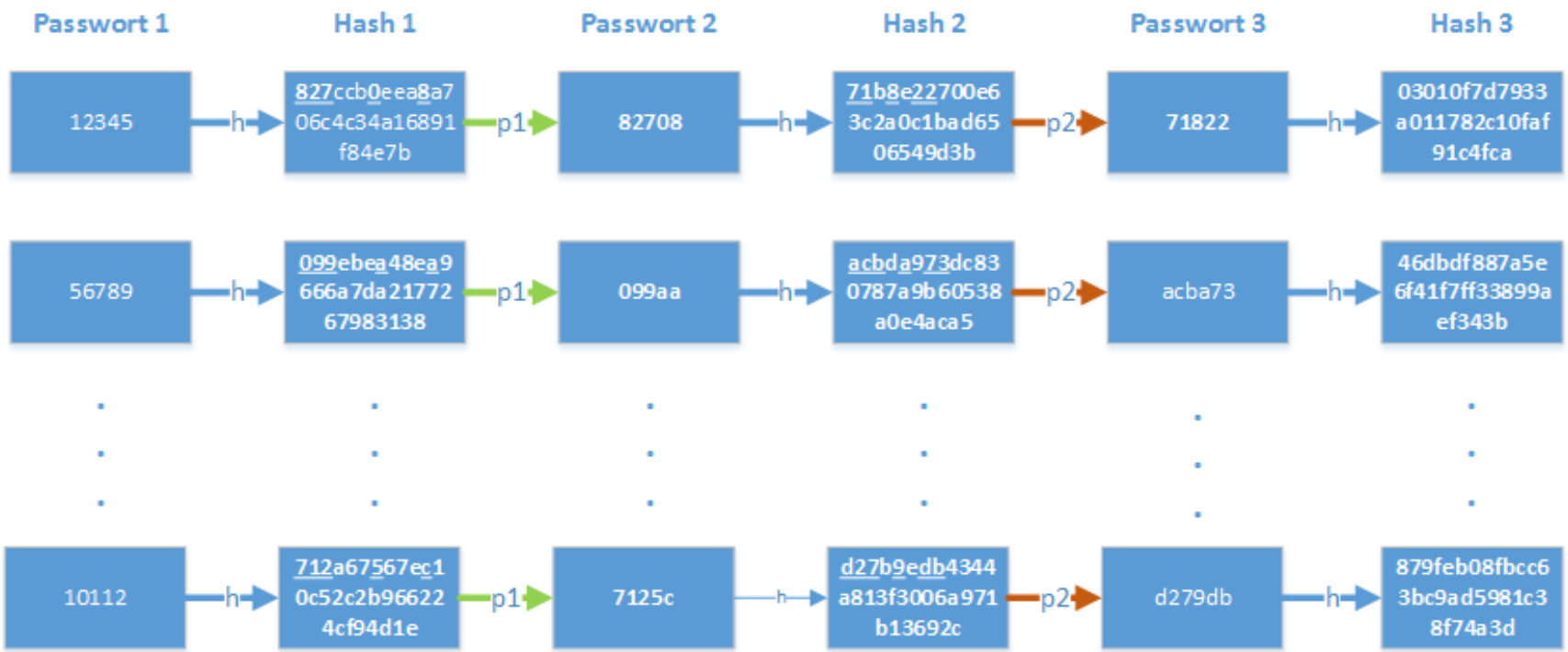


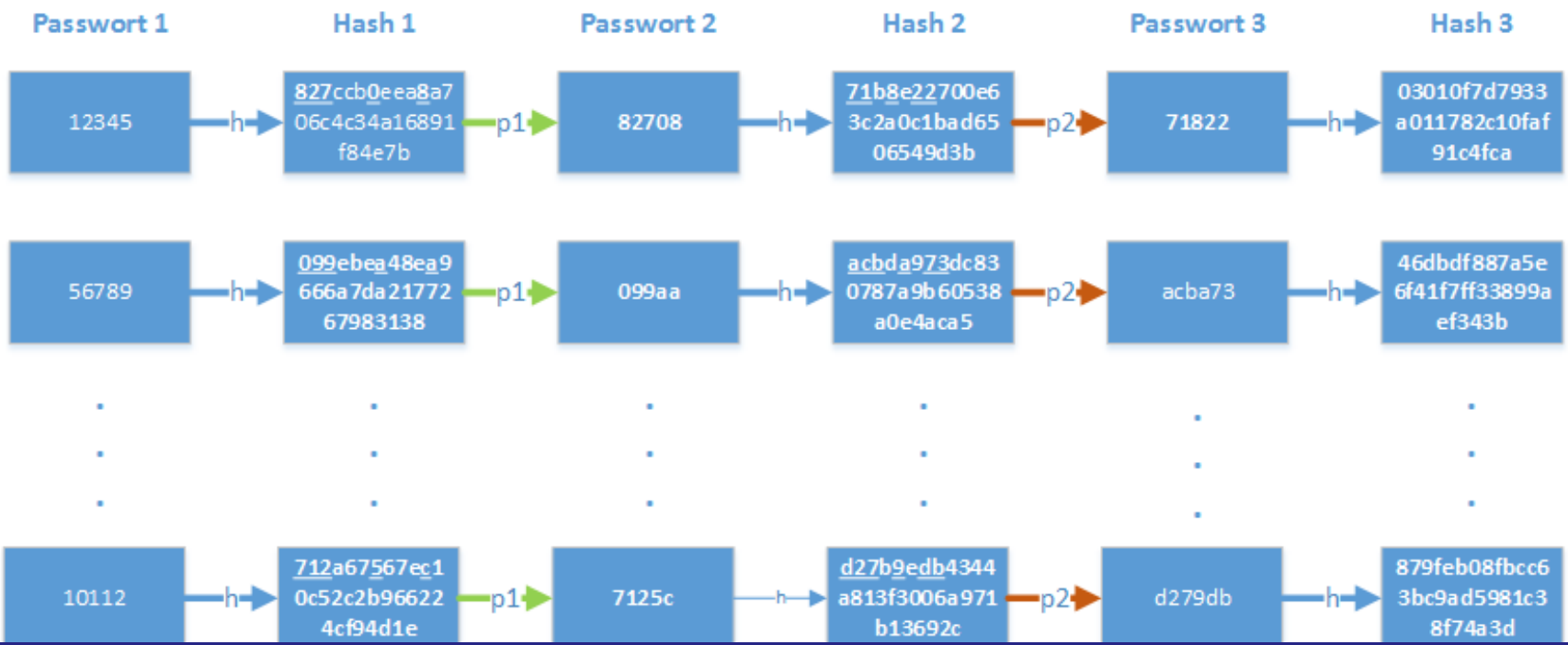
Abbildung: In Anlehnung an [Sorge2013]

Rainbowtables - Berechnung

- Wir haben einen Hash, z.B. aus der */etc/shadow*
- **Annahme:** Der Hash befindet sich in der letzten Spalte → Starte mit der letzten Reduktionsfunktion
- **Annahme:** Der Hash befindet sich in der vorletzten Spalte → Starte mit der vorletzten Reduktionsfunktion
- ...
- Ist der Hash gefunden, rekonstruiere die Kette

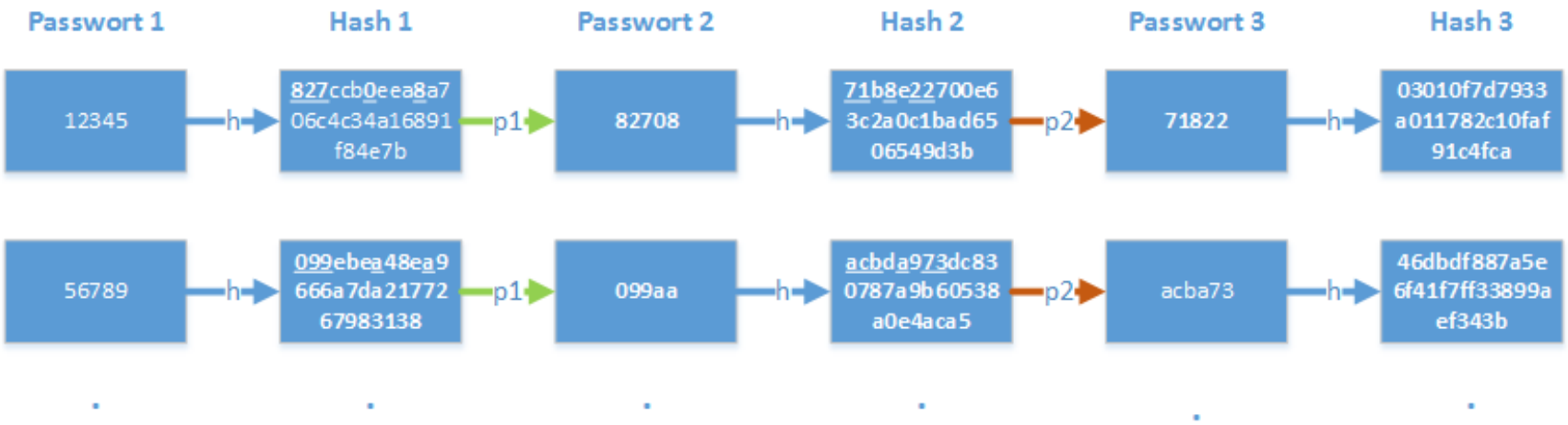
Rainbowtables - Beispiel (I)

- Wir haben den Hash
  $h(x) = 099ebea48ea9666a7da2177267983138$ aus der
  /etc/shadow



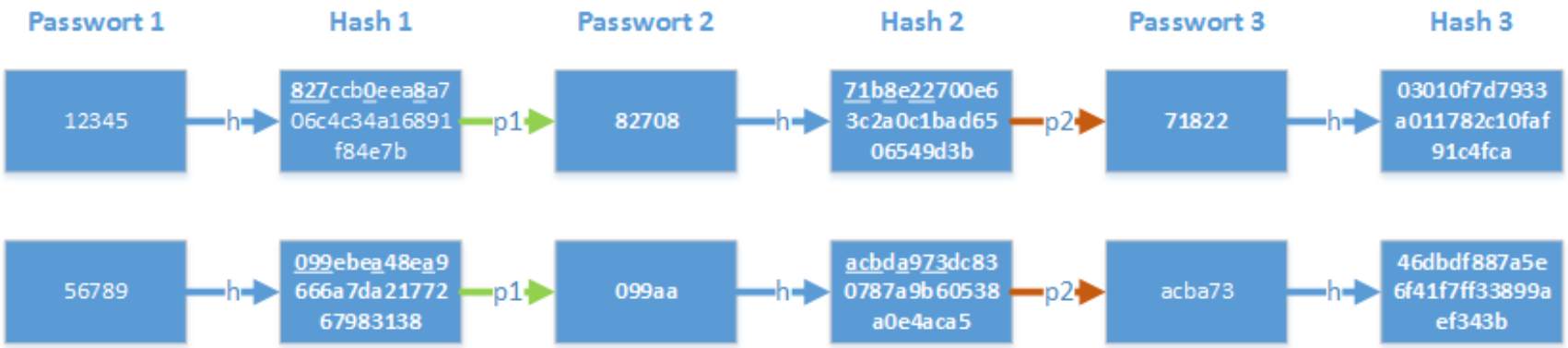
Rainbowtables - Beispiel (II)

- Ist der Hash in der letzten Spalte? $\Rightarrow$ Nein!
- Wir reduzieren den Hashwert auf ein anderes Passwort.
  $P_2(099ebea48ea9666a7da2177267983138) = 09eea$
- Wir berechnen
  $h(09eea) = c0008ec5eb3461e63f399bd8d9210474$
- Ist der Hash in der letzten Spalte? $\Rightarrow$ Nein!



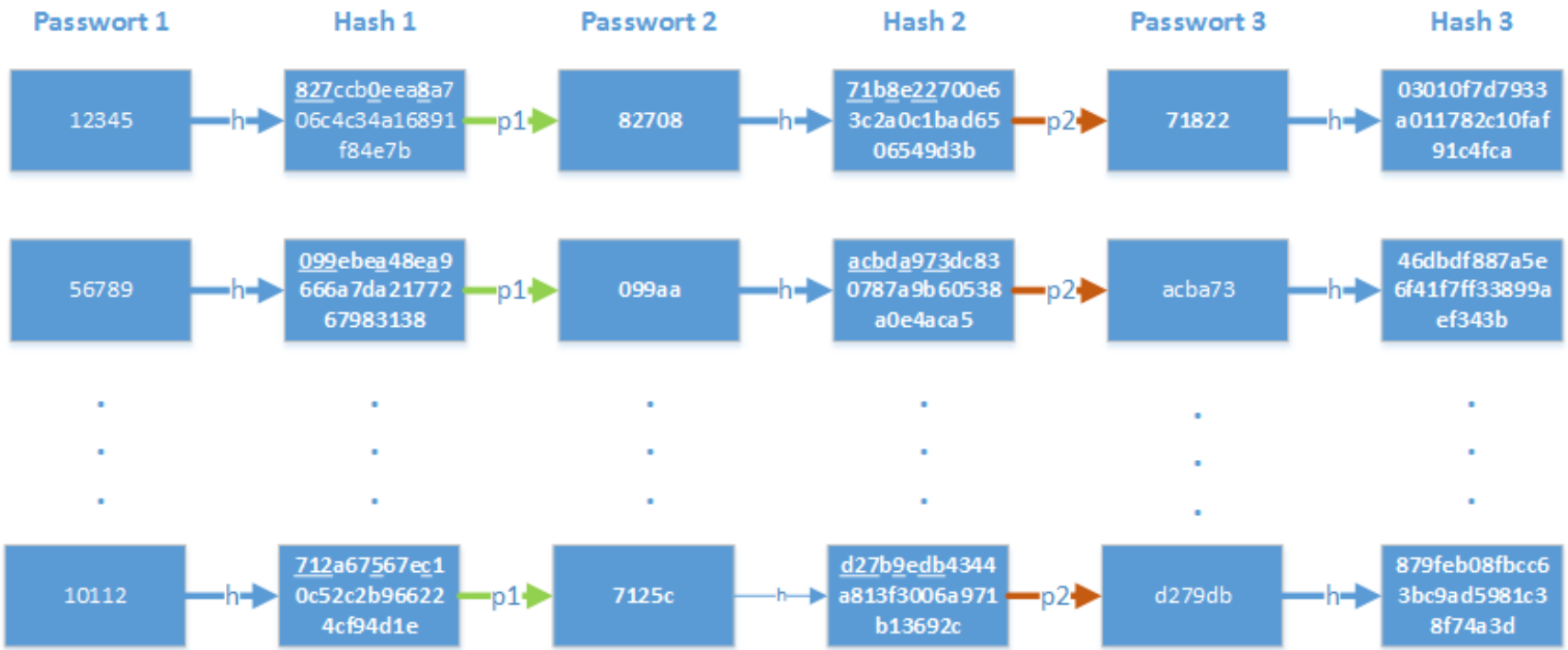
Rainbowtables - Beispiel (III)

- Wir reduzieren den Hashwert auf ein anderes Passwort.
  P_1(099ebea48ea9666a7da2177267983138) = 099aa
- Wir berechnen
  $h(099aa) = acbda973dc830787a9b60538a0e4aca5$
- P_2(acbda973dc830787a9b60538a0e4aca5) = acba73
- h(acba73) = 46dbdf887a5e6f41f7ff33899aef343b



Rainbowtables - Beispiel (IV)

- Ist der Hash in der letzten Spalte? $\Rightarrow$ Ja!
- Wir haben die Kette $P_1 \rightarrow h \rightarrow P_2 \rightarrow h$ berechnet



Rainbowtables - Grafik

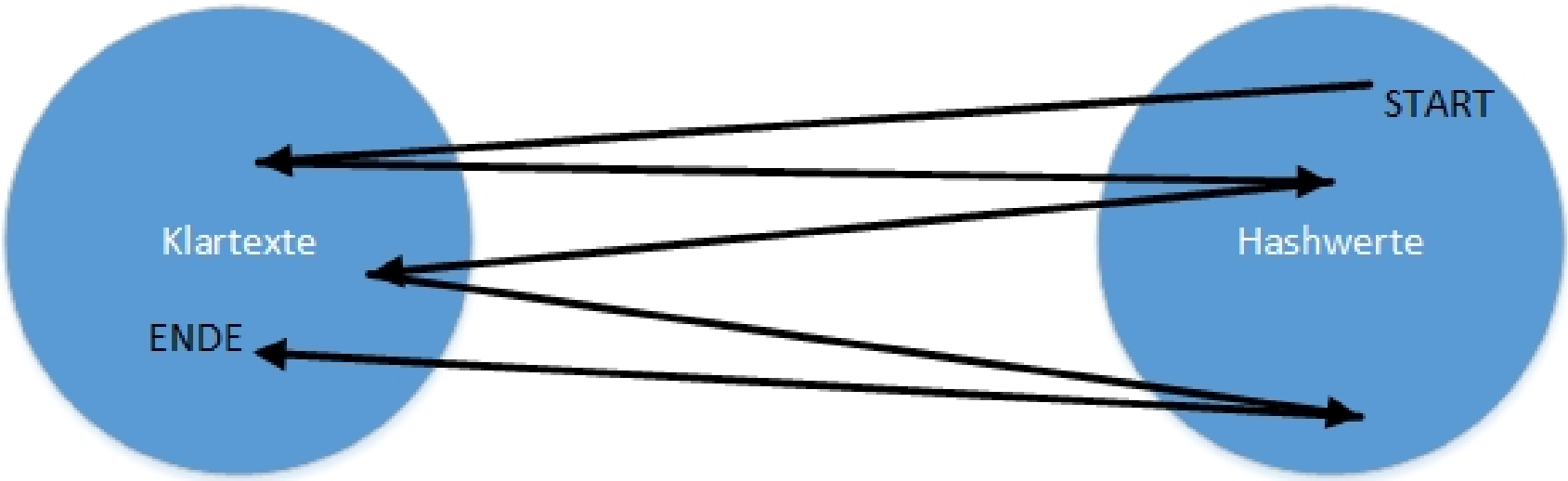


Abbildung: In Anlehnung an [Kuliukas]

Rainbowtables - Beschaffung

- Web Services
- Projekte
- Diverse Downloads
- Selbst berechnen

Rainbowtables - Probleme

- Die Reduktion erfasst nicht alle Passwörter $\Rightarrow$ 100% Erfolg ist nicht möglich.
- Extremer Aufwand bei längeren Passwörter
  - Tabelle wird sehr groß, wenn „alles“ abgedeckt werden soll
  - Berechnung dauert entsprechend lange
  - Viele Möglichkeiten
- Es gibt Gegenmaßnahmen

Gegenmaßnahmen (I)

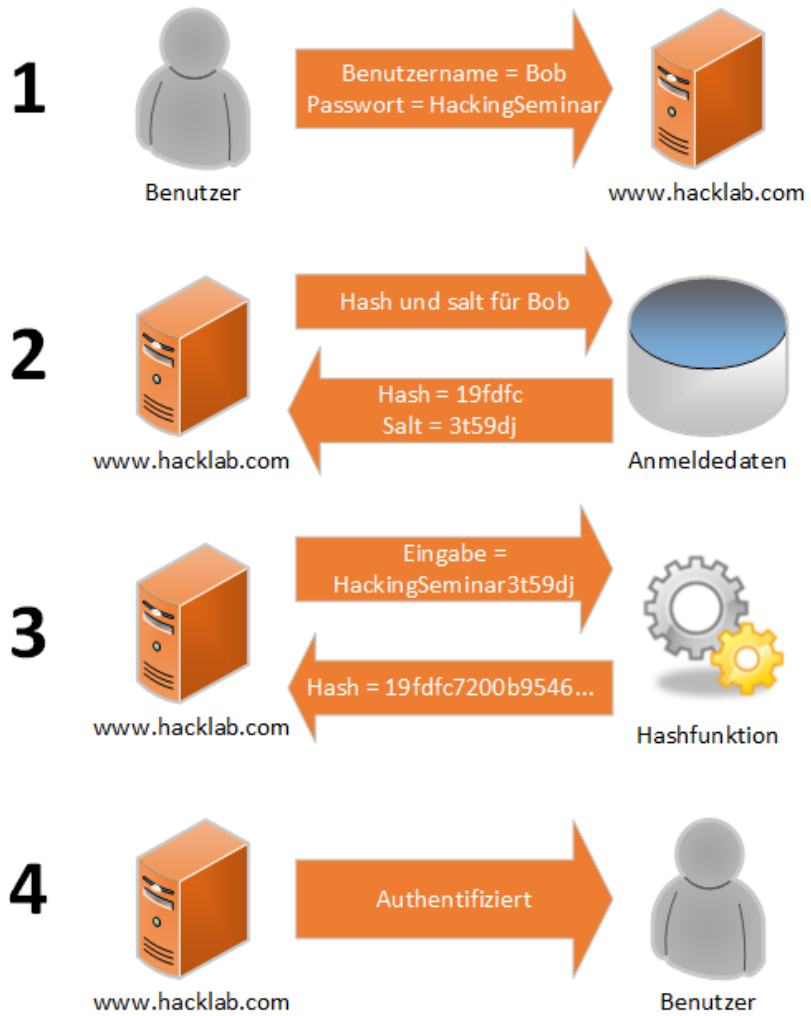
- Einsatz von **Salts**
  - Passwort wird mit einem zufälligen Wert (Salt) verlängert
    - Idealerweise sollte das Salt für jeden Benutzer unterschiedlich sein
    - Selbst wenn das Salt überall gleich ist, braucht der Angreifer immer noch einen neuen Rainbowtable
  - Speicherung der folgenden Informationen user,id,salt,hash($salt \| passwort$) - Salt auch im Klartext
  - Angreifer benötigt Rainbowtable für jeden Salt-Wert

Gegenmaßnahmen (II)

Mehrfaches Anwenden von Hashfunktionen

- Verlangsamt sowohl den Angreifer, als auch die Validierung.
- Hashes können unter normalen Umstanden sehr schnell berechnet werden $\rightarrow$ vernachlässigbar
- Die zusätzliche Zeit für die Generierung der Rainbowtables kann jedoch hoch werden.
- Prüfung: 1 Mikrosekunden $\rightarrow$ 1 Millisekunde

Authentifizierung



Quelle: Anlehnung an [MSDLT]

Angriffe auf Hashes und Rainbowtables

- Hashcat
  - MD5
  - SHA1
- Online Tools - Für kurze Klartexte
  - http://www.md5decrypter.co.uk/
  - http://www.md5decrypter.co.uk/sha1-decrypt.aspx
- Rainbowtables
  - ophcrack
  - https://www.objectif-securite.ch/en/ophcrack.php
  - Rainbow Crack
  - ElcomSoft Brute Forcers
  - Free Rainbow Tables Project

Danke für die Aufmerksamkeit

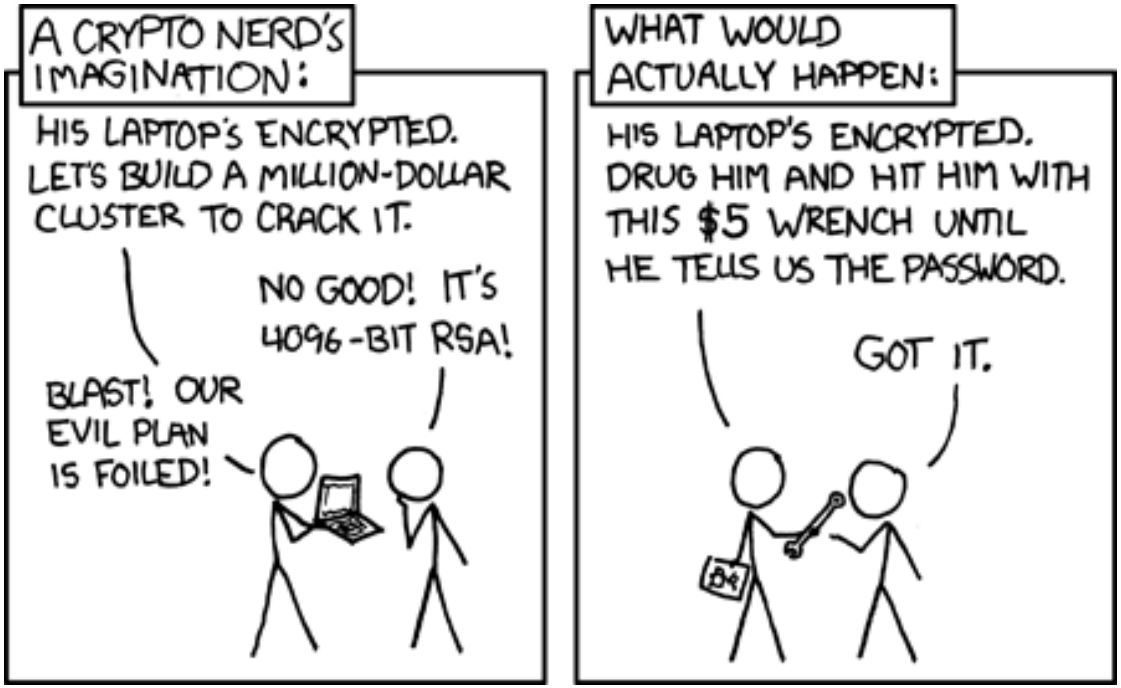


Abbildung: Quelle: *http* : *//xkcd.com/538/*

Quellen I


Bundesamt für Sicherheit in der Informationstechnik.
Passwörter, 11 2013.
URL https://www.bsi-fuer-buerger.de/BSIFB/DE/MeinPC/Passwoerter/passwoerter_node.html.


M.E. Hellman.
A cryptanalytic time-memory trade-off.
Information Theory, IEEE Transactions on, 26(4):401–406, 1980.
ISSN 0018-9448.
doi: 10.1109/TIT.1980.1056220.

Quellen II


Jonathan Katz and Yehuda Lindell.
Introduction to modern cryptography.
Chapman Hall CRC, Boca Raton [u.a.], 2008.
ISBN 978-1-58488-551-1, 9781584885511, 1584885513, 1-58488-551-3.
URL http://ubprp2/records/PAD_ALEPH000975303.


Kestas Kuliukas.
How rainbow tables work, 11 2013.
URL http://kestas.kuliukas.com/RainbowTables/.

Quellen III

Philippe Oechslin.
Making a faster cryptanalytic time-memory trade-off.
In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630. Springer, 2003.
doi: 10.1007/978-3-540-45146-4_36.
URL http://www.iacr.org/cryptodb/archive/2003/CRYPTO/1615/1615.pdf.

Prof. Dr. rer. nat. Johannes Blömer.
Hashfunktionen und authentifizierungscodes, 2013.

Quellen IV

Jun.-Prof. Dr. Christoph Sorge.
Chapter 5: (local) identification and authentication.
a, 2013.

Secure Developer Lifecycle Team.
Secure credential storage, 1 2012.
URL http://blogs.msdn.com/b/sdl/archive/2012/01/16/secure-credential-storage.aspx.

Dennis Elser und Micha Pekrul.
Inside the password-stealing business: the who and how of identity theft.
Technical report, McAfee Avert Labs, 2009.

Quellen - Folie: „Passwortklau“

- Vudu: http://heise.de/-1838222
- Adobe: http://heise.de/-2048014
- Simyo: http://heise.de/-2054990
- GitHub: http://heise.de/-2051046

Beispielhafte Anwendung von Hashfunktionen

- Alice möchte Daten speichern, sodass sie vor Veränderungen geschützt sind, bzw. das eine Veränderung entdeckt werden kann (Integrität).
- Alice besitzt großen und unsicheren Speicher und eine Smartcard mit kleinem, aber sicheren Speicher
- Alice wählt eine kollisionsresistente Hashfunktion
- Alice Speichert die Daten x auf dem großen Speicher und $h(x)$ auf der Smartcard

Beispielhafte Anwendung von Hashfunktionen

- Angreifer möchte die Daten auf dem großen Speicher ändern.
- Problem: Alice benutzt kollisionsresistente Hashfunktion
- Angreifer findet kein x_2, sodass $h(x_1) = h(x_2)$
- Alice würde bemerken, dass die Daten geändert wurden.

Angriffe auf Hashfunktionen

- Brute-Force
- Geburtstagsattacke
- Yuvals Angriff

Bekannte Hashfunktionen

- BruteMessag-Diget Familie (MD4 / MD5)
  - **MD5:** Bereits 1996 Kollisionen gefunden und 2005 erste x.509 Zertifikate gefälscht
  - Sollten nicht genutzt werden
  - Wird noch oft genutzt (z.B. Wordpress).
- Secure Hash Familie (SHA1)
  - **SHA-1:** Angriffe bekannt → „Word Expansion“ - Hat Schwachstellen, kann aber noch problemlos genutzt werden.
  - **SHA-2:** Gilt als sicher
  - **SHA-3:** Aktuell in der Standardisierung. Gerüchte über Arbeiten der NSA.

Authentifizierung - Besitz

- (Kryptographischer) Schlüssel
- Chipkarten
- Personalausweise
- ...
- Nachteile
  - Verlieren
  - Klauen
  - Duplizieren

Authentifizierung - Biometrie

- Fingerabdruck
- Iriserkennung
- Gesichtserkennung
- „Schrift“ einer Person
- ...
- Nachteile
  - Fehlerhafte Erkennung möglich
  - Spezielle Technik notwendig
  - Nicht unbedingt fälschungssicher
    - Fälschung von Fingerabdrücken (Beispiel: Apples TouchID)
    - Android Gesichtserkennung
    - Verbreitung nicht gesichert → z.B. Menschen ohne Fingerabdruck

Authentifizierung - Zwei-Faktor-Authentifizierung

- Kombination von Authentifizierungsmechanismen
  - **Beispiel:** Bank
  - Besitz + Wissen = Zugang
  - EC-Karte + Pin = Zugang
- Kann die Sicherheit der Verfahren erhöhen

Geburtstagsangriff (I)

- Frage: Wie groß ist die Wahrscheinlichkeit, dass von beliebig ausgewählten Personen 2 Personen am gleichen Tag Geburtstag haben?
- Wahrscheinlichkeit, dass keine zwei Personen mit dem gleichen Geburtstag in einer Gruppe von k Personen vorhanden sind: (365-k)/365.
- Gruppe mit einer Person: $p_1 = \frac{365}{365} = 1$
- Gruppe mit zwei Personen: $p_2 = p_1 \cdot \frac{364}{365} \approx 0.99$
- Gruppe mit drei Personen: $p_3 = p_2 \cdot \frac{363}{365} \approx 0.99$
- ...
- Gruppe mit n Personen: $p_n = (365/365) \cdot (364/365) \cdot \ldots \cdot (365 - n/365)$

Geburtstagsangriff (II)

- $n = 22 \Rightarrow p_{22} = p_{21} \cdot \frac{365-22}{365} \approx 0,52430469$
- $n = 23 \Rightarrow p_{23} = p_{22} \cdot \frac{365-23}{365} \approx 0,50729723$
- Gruppe mit 23 Personen $\rightarrow$ Wahrscheinlichkeit größer als 50%
- Für n Personen gibt es $\frac{n \cdot (n-1)}{2}$ verschiedene Paare, die am gleichen Tag Geburtstag haben könnten.

Geburtstagsangriff (III)

- Konsequenzen für Hashes
- Beispiel: 64 Bit Hashwert $\rightarrow \approx 2^{32} \Rightarrow 2^{\frac{k}{2}}$ Versuche
- Folgerung: k möglichst groß

Sichere Passwörter - Twitter

- Mindestens 20 Zeichen
- Speichern mit Passwortmanager
- Eigenes System nur für die Twitternutzung.

Sichere Passwörter - BSI

- Mindestens acht Zeichen lang
- Groß- und Kleinbuchstaben sowie Sonderzeichen und Ziffern
- Nicht in Wörterbüchern vorkommen
- Keine Tastaturmuster
- Passwörter nicht notieren
- Passwörter regelmäßig ändern
- Keine einheitlichen Passwörter
- Voreingestellte Passwörter ändern
- Passwörter nicht versenden

Schleifen - Hellmann

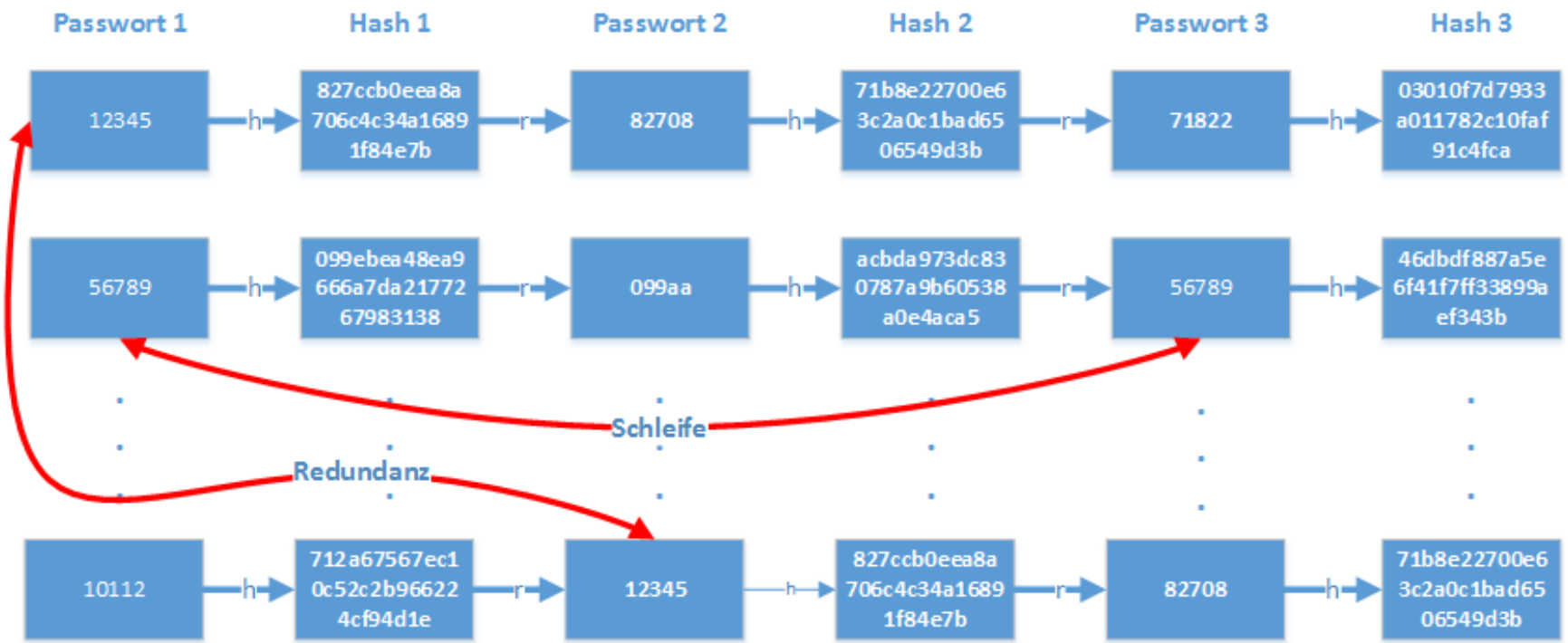


Abbildung: In Anlehnung an [Sorge2013]

Rainbowtable - Größe

SHA1 Kleinbuchstaben + Zahlen	1 bis 9 Zeichen	80 GB
SHA1 Kleinbuchstaben + Zahlen	1 bis 10 Zeichen	396 GB
SHA1 Mixalpha + Zahlen	1 bis 9 Zeichen	864 GB

- Quelle: http://project-rainbowcrack.com
- Größe stark von der Länge der Kette abhängig