Übung GSS Blatt 1

SVS – Sicherheit in Verteilten Systemen

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Organisatorisches

- Den Foliensatz mit Hinweisen zur Organisation (in STINE) lesen!
- Termine für Übung/Vorlesung
  - Grundsätzlich: STINE-Nachricht bei kurzfristigen Änderungen
  - Bei zu starker bzw. zu geringer Auslastung können sich Räume zum 2. Übungstermin noch einmal ändern. Vorher informieren!
- 3er–5er Gruppen: Wer noch keine Gruppe hat, schließt sich beim 1. Termin einer existierenden Gruppe an; notfalls terminübergreifende Gruppen bilden.
- Fragen zum IS-Teil bitte an euren Übungsgruppenleiter oder in dringenden Fällen an Ephraim Zimmer (ezimmer@informatik.uni-hamburg.de) stellen.
- Bestehen der GSS-Übung:
  Im Mittel müssen auf den jeweiligen Aufgabenblättern im OSDS-Teil und im IS-Teil mind. 50 % erreicht werden.
- Quellen sind anzugeben! Plagiate werden mit 0 Punkte bewertet!!!

Aufgabe 1: Allgemeine Aussagen zur IT-Sicherheit

- Teilaufgabe 3: Ursachen
  - Nennen Sie die Ihrer Meinung nach drei häufigsten Ursachen für mangelnde IT-Sicherheit in Unternehmen. Wo finden Sie belastbare Referenzen für Ihre Vermutungen?

Aufgabe 1: Allgemeine Aussagen zur IT-Sicherheit

- Teilaufgabe 3: Ursachen
  - Nennen Sie die Ihrer Meinung nach drei häufigsten Ursachen für mangelnde IT-Sicherheit in Unternehmen. Wo finden Sie belastbare Referenzen für Ihre Vermutungen?

- Aus der Microsoft-Sicherheitsstudie 2010:

a)	Es fehlt an Bewusstsein bei den Mitarbeitern	59%
b)	Es fehlt an Geld/Budget	57%
c)	Es fehlt an Bewusstsein beim mittleren Management	54%
d)	Es fehlt an Bewusstsein und Unterstützung im Top-Management	47%
e)	Es fehlen verfügbare und kompetente Mitarbeiter	41%
f)	Die Kontrolle auf Einhaltung ist unzureichend	38%
g)	Es fehlt an Möglichkeiten zur Durchsetzung sicherheitsrelevanter Maßnahmen	35%
h)	Es fehlen strategische Grundlagen / Gesamt-Konzepte	31%
i)	Anwendungen sind nicht für IS-Maßnahmen vorbereitet	27%

Basis: 133 Antworten

<kes>/Microsoft-Sicherheitsstudie 2010

Aufgabe 3: Angreifermodell

- Teilaufgabe 1
  - Was ist ein Angreifermodell und wie definiert man es?

Aufgabe 3: Angreifermodell

- Teilaufgabe 1
  - Was ist ein Angreifermodell und wie definiert man es?
  - Das Angreifermodell definiert die maximal berücksichtigte Stärke eines Angreifers, gegen den ein Schutzmechanismus gerade noch wirkt.

Aufgabe 3: Angreifermodell

- Teilaufgabe 1
  - Was ist ein Angreifermodell und wie definiert man es?
  - Das Angreifermodell definiert die maximal berücksichtigte Stärke eines Angreifers, gegen den ein Schutzmechanismus gerade noch wirkt.
  - Es berücksichtigt die:
    - Rollen,
    - Verbreitung,
    - Verhalten (aktiv, passiv) sowie
    - Rechenkapazität des Angreifers.

Aufgabe 5: Praktischer Teil – Passwörter

- Teilaufgabe 2: Einfaches Hash-Verfahren
  - *Hashfunktionen:* Durch Hashfunktionen können Kennwörter sicherer als im Klartext hinterlegt werden. Wie funktionieren Kennwortspeicherung und -prüfung bei diesem Verfahren?

Aufgabe 5: Praktischer Teil – Passwörter

- Teilaufgabe 2: Einfaches Hash-Verfahren
  - *Hashfunktionen:* Durch Hashfunktionen können Kennwörter sicherer als im Klartext hinterlegt werden. Wie funktionieren Kennwortspeicherung und -prüfung bei diesem Verfahren?

- $h: M \rightarrow H$ mit $|M| >= |H|$
- **Kryptographische Hashfunktionen:** ermöglichen es, einen Inhalt nahezu eindeutig zu identifizieren, ohne den Inhalt zu verraten.
- Entscheidend für Passworthinterlegung: **Urbild-Resistenz** (engl. pre-image resistance), d.h. bei gegebenem Hashwert $H_1=h(p_1)$ soll es schwierig sein, ein Passwort p_2 zu finden mit $h(p_2)=H_1$
- Das **Finden eines Urbildes** einer kryptographischen Hash-Funktion zu einem gegebenen Hashwert H_1, d.h. hier das Ermitteln des zugehörigen Passworts p_1 oder eines anderen Passworts p_2, welches den gleichen Hash erzeugt, ist ein „schwieriges" Problem.

- Speicherung: Hashen des Passworts, Abspeichern des Hashwerts
- Überprüfung: Hashen des Passworts, Überprüfung der Hashwerte

Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

- Time-Memory-Trade-Off

  - Um das Passwort zu einem Hashwert ad hoc zu ermitteln, ist es notwendig, alle in Frage kommenden Passwörter auszuprobieren (d.h. die Hashfunktion darauf anzuwenden)
    - hoher (linearer) Zeitaufwand für Suche
    - kein Speicherbedarf
    - „Brute-Force-Attack“

  - Alternative: alle möglichen Passwörter vorab einmalig hashen; Hashwert zusammen mit dem zugehörigen Passwort in einer Datenbank (Lookup-Tabelle, sog. „Dictionary“) zu speichern.
    - Speicherbedarf: hoch (linear)
    - Zeitaufwand für Suche: gering (sublinear, z.B. log(n))
    - „Dictionary-Attack“

Die folgenden Folien zu Rainbow Tables nach: Florian Scheuer, Uni-R., 2010

Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

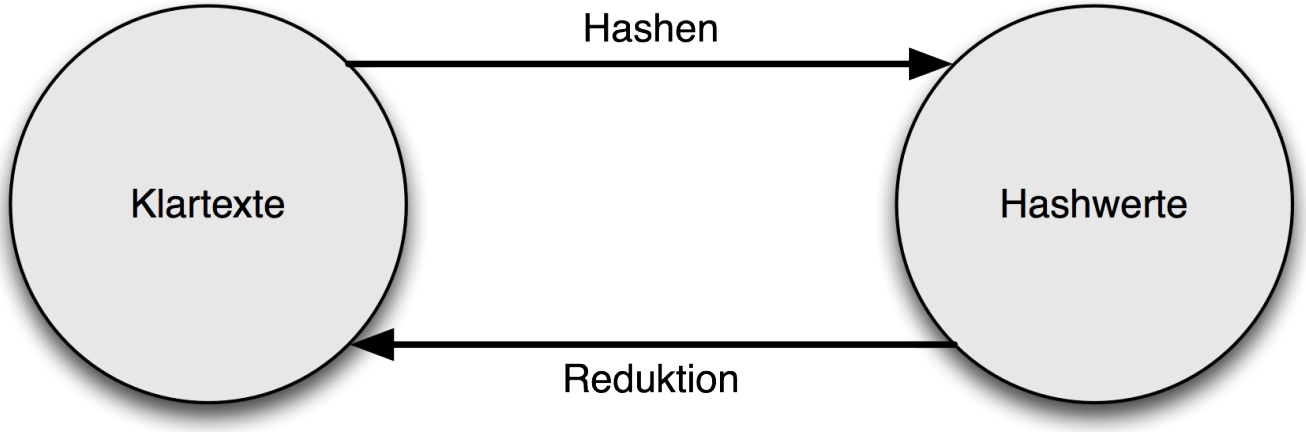
- Time-Memory-Trade-Off
  - Erfinder der „Rainbow Tables“ ist Philippe Oechslin
  - Rainbow Tables erlauben
    - die Zeit zur Ermittlung eines Passworts anhand eines gegebenen Hashwerts im Vergleich zum **Brute-Force-Angriff** zu reduzieren

    und gleichzeitig

    - weniger Speicherplatz zu benötigen als ein vollständiges **Dictionary**
  - Die „Stellschraube“ zwischen Zeit- und Speicherbedarf lässt sich dabei beliebig verschieben.

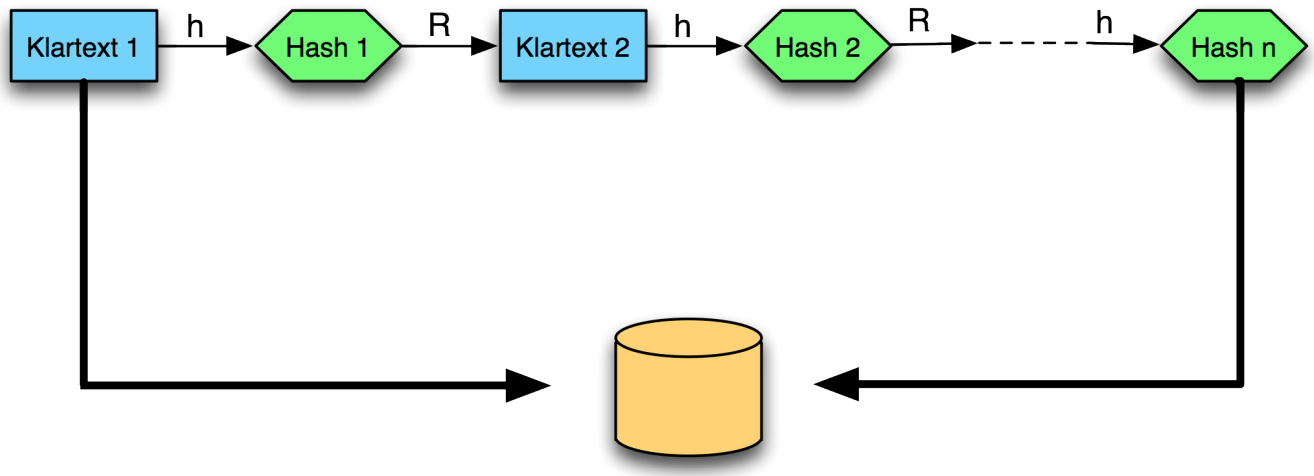
Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

- Hash- und Reduktionsfunktionen
  - Hashfunktion erzeugt aus einem Passwort einen Hashwert
  - Reduktionsfunktion bildet einen Hashwert auf ein Passwort aus dem Raum aller möglichen Passwörter (für gegebenen Zeichenvorrat) ab



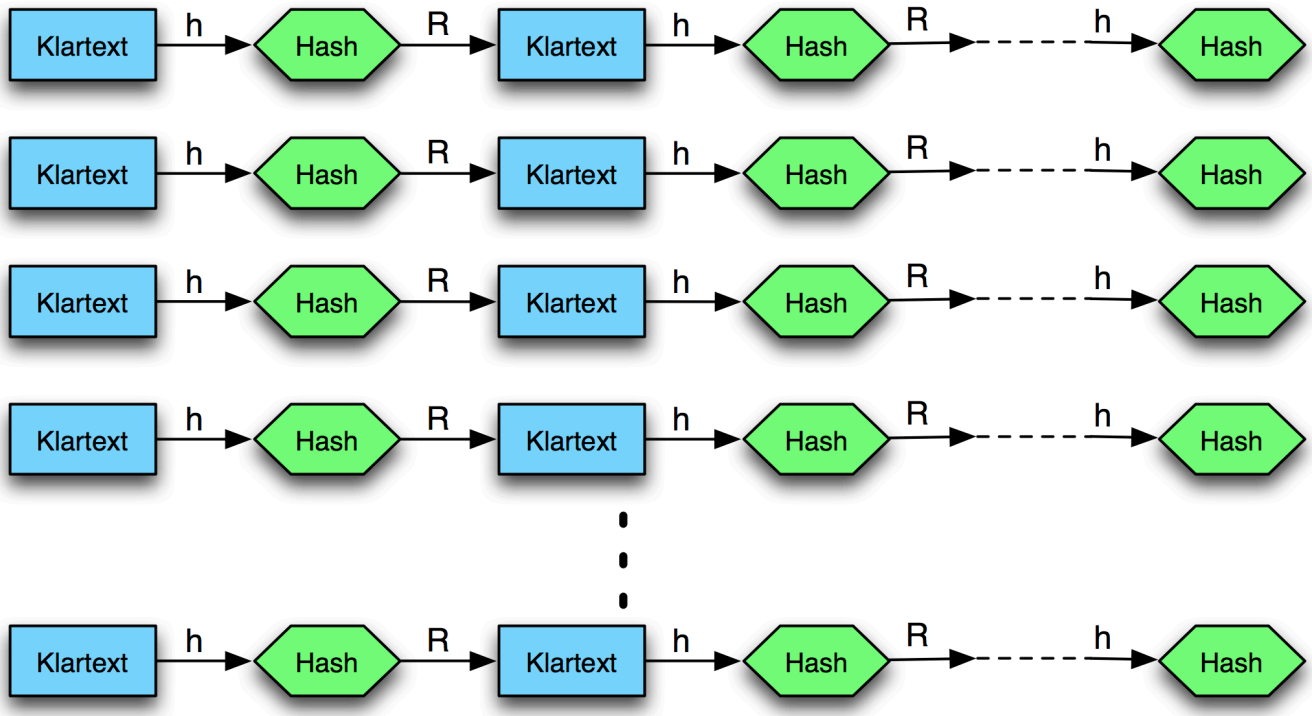
Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

- Generierung einer Kette
  - Wahl eines zufälligen Klartextes (Passwort)
  - *n*-malige Anwendung von Hash- und Reduktionsfunktion
  - Hinzufügen eines Eintrags in das Dictionary (auf der Festplatte)
    - Schlüssel: Hashwert *n*
    - Wert: Klartext 1



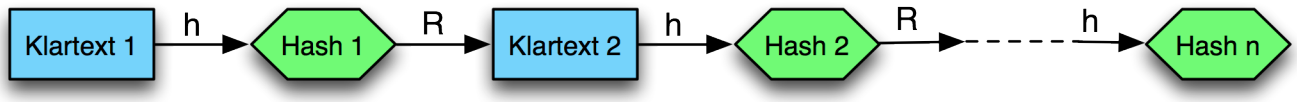
Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

- Warum der Begriff einer Tabelle?
  - Berechnung und Speicherung einer großen Anzahl von Ketten
  - Stellschraube für den Trade-Off: Kettenlänge, Anzahl der Ketten



Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

- Algorithmus zur Anwendung von Rainbow-Tables



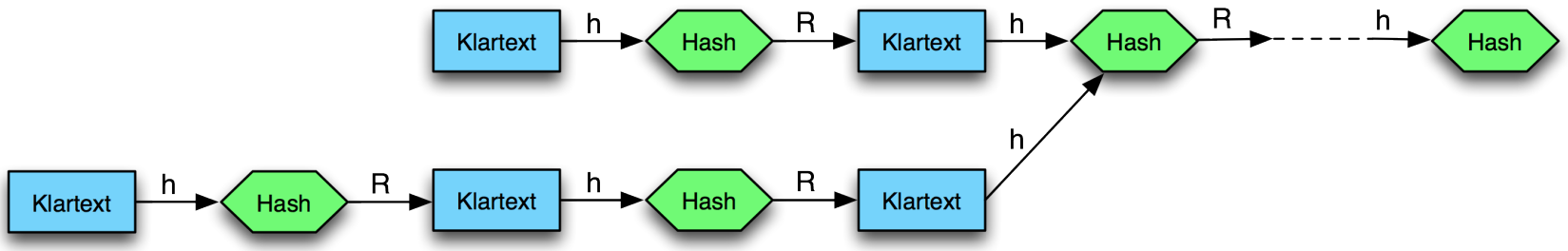
1. Gegebenen Hashwert in der Liste der gespeicherten Hashwerte (rechte Spalte in der Table) nachschlagen.
   Treffer? gehe zu Schritt 3; sonst: gehe zu Schritt 2.

2. Auf den Hashwert die Reduktions- und dann wieder die Hashfunktion anwenden, um neuen Hashwert zu erhalten. Diesen erneut nachschlagen.
   Treffer? gehe zu Schritt 3; sonst: gehe zu Schritt 2.

3. Anfangs-Klartext (linke Spalte) der gefundenen Kette nehmen und so lange abwechselnd hashen und reduzieren, bis man den gegebenen Hashwert erhält. Das zug. Klartextpasswort kann unmittelbar „links“ neben dem gerade erzeugten Hashwert abgelesen werden.

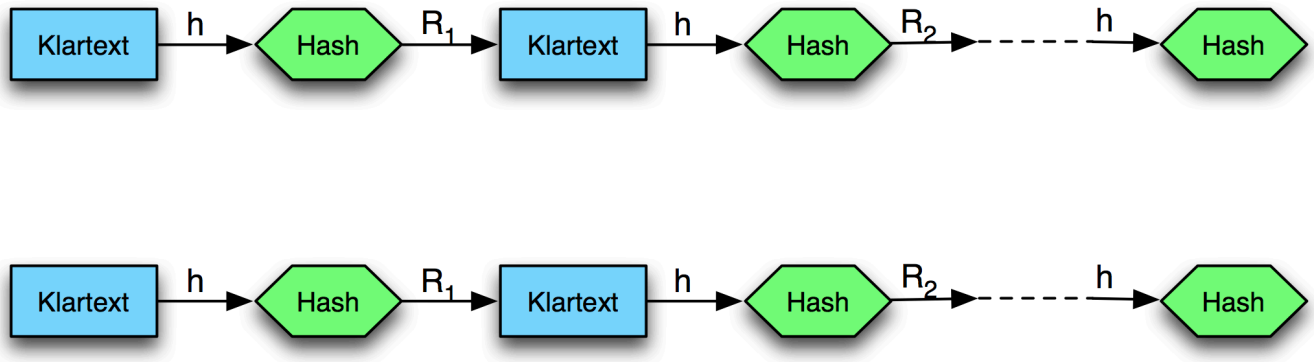
Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

- Problem: Merge
  - Tritt an irgendeiner Stelle einer Kette ein Wert Klartext auf, der auch in einer anderen Kette auftritt, vereinigen sich beide Ketten (Merge).
  - Für einen Hash-Wert kann jedoch nur ein Klartext abgespeichert werden, d.h. die später generierte Kette kann nicht zur Tabelle hinzugefügt werden.
  - Auswirkung: Alle auf der später generierten Kette befindlichen Passwörter können mit dieser Tabelle nicht gefunden werden.



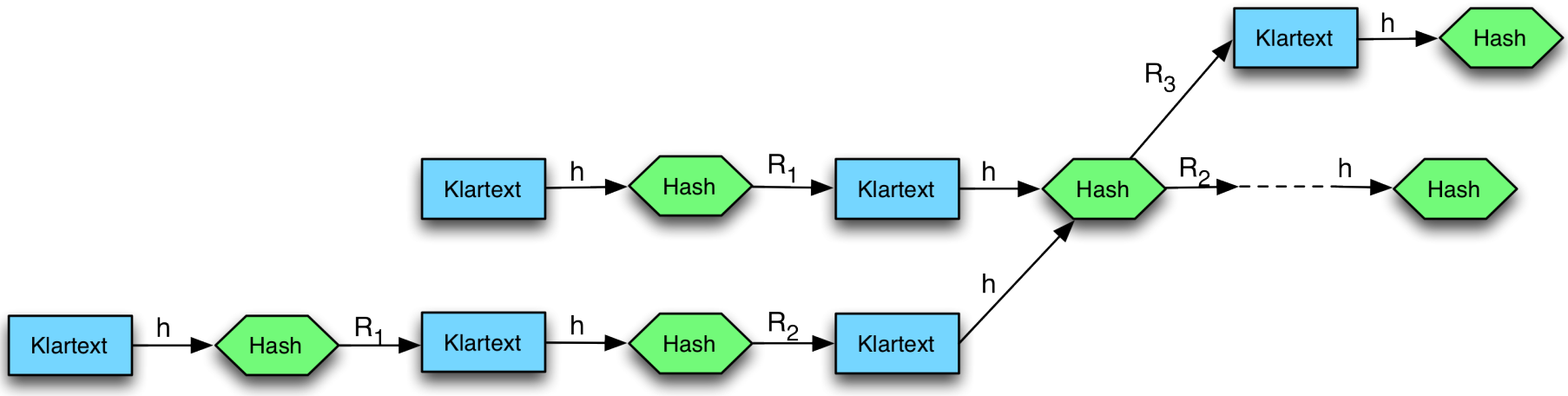
Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

- Lösung: unterschiedliche Reduktionsfunktionen
  - Bei der Generierung wird in jeder Iteration eine andere Reduktionsfunktion verwendet, jedoch über alle Ketten hinweg immer dieselbe.



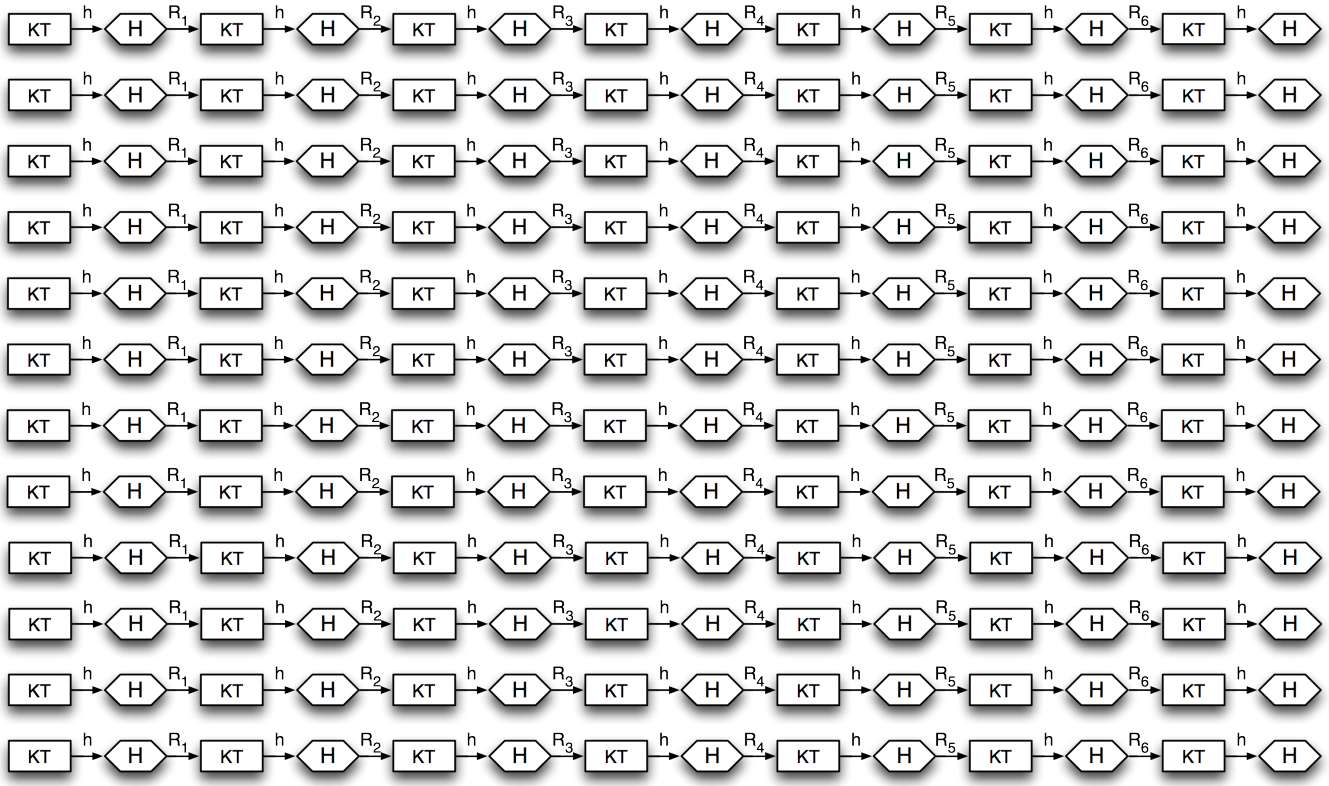
Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

- Lösung: unterschiedliche Reduktionsfunktionen
  - Vereinigungen treten jetzt nur noch auf, wenn derselbe Klartext in zwei Ketten in **genau derselben Iteration** auftritt (wesentlich unwahrscheinlicher)
  - Andernfalls treffen sich die Ketten lediglich punktuell und divergieren im weiteren Verlauf wieder.



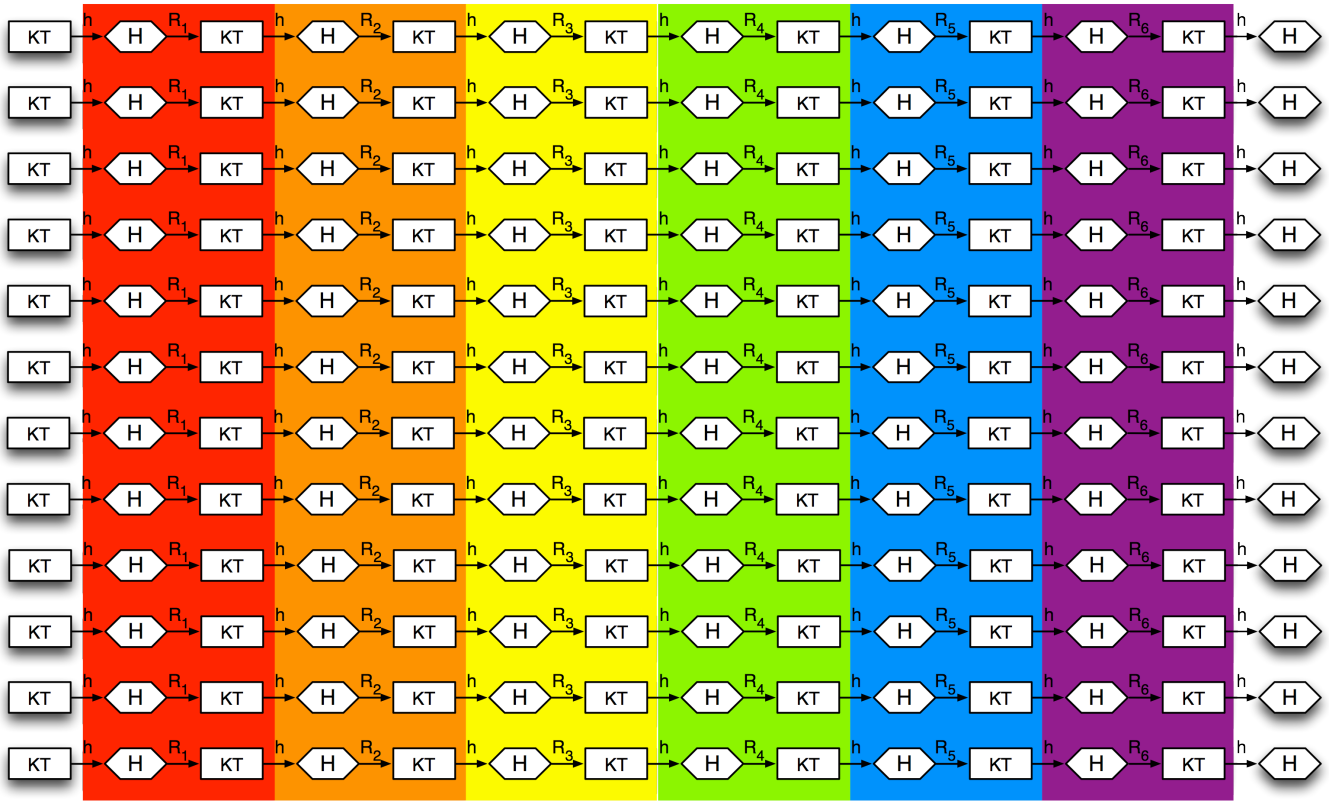
Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

- Warum Regenbogen?



Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

- Darum ☺



Aufgabe 5.4: Passwörter mit Rainbow-Tables knacken

- Grenzen der Rainbow-Tables
  - Durch die Form der Erstellung ist es extrem unwahrscheinlich, dass **jedes** mögliche Passwort enthalten ist.
  - Eine Aussage ist nur noch auf Basis von Wahrscheinlichkeiten zu treffen.
  - Die verbreitetsten Rainbow-Tables weisen nach eigenen Angaben eine Abdeckung zwischen 85% und 99.9% der möglichen Passwörter auf
    (z.B. https://www.freerainbowtables.com/en/tables2/)

Nachsatz: Dictionary-Attack != Dictionary-Attack

- Mit dem Begriff Dictionary-Attack werden im Zusammenhang von Angriffen auf gehashte Passwörter zwei verschiedene Angriffe bezeichnet...

1. Brute-Force-Angriff mit einem Dictionary
   - Dictionary enthält Terme einer Sprache
   - Ziel: Reduktion der Anzahl der durchzuprobierenden Passwörter

Nachsatz: Dictionary-Attack != Dictionary-Attack

- Mit dem Begriff Dictionary-Attack werden im Zusammenhang von Angriffen auf gehashte Passwörter zwei verschiedene Angriffe bezeichnet...

1. Brute-Force-Angriff mit einem Dictionary
   - Dictionary enthält Terme einer Sprache
   - Ziel: Reduktion der Anzahl der durchzuprobierenden Passwörter

2. Vorberechnung und Speicherung von Hashwert/Passwort-Paaren
   - Dictionary ist das persistent gespeicherte Ergebnis des Durchprobierens (Hashens) aller möglichen Passwörter
   - Sortierte Tabelle von Hash-Werten und zug. Passwörtern
   - Ziel: „unmittelbares" Nachschlagen eines Passworts ohne jedes Mal wieder den gesamten Möglichkeitenraum durchzuprobieren