Fingerprints und Hashes

https://iuk.one/DS-FINGER.pdf

Clemens H. **Cap**
ORCID: 0000-0003-3958-6136

Department of Computer Science
University of **Rostock**
Rostock, Germany
clemens.cap@uni-rostock.de

2026-11-23 18:09



Ziel

(Kryptographische) Hash-Funktionen sind **eine Art Fingerabdruck**
ganz ähnlich dem Fingerabdruck eines Menschen.

- Man kann das Dokument nicht ändern, ohne den Fingerabdruck stark zu ändern.
- Aus dem Fingerabdruck kann man das Dokument nicht rekonstruieren.
- Der Fingerabdruck ist eine hinreichend gute **Charakterisierung** des Dokuments in dem **folgenden anschaulichen Sinn**:
  1. Es ist unwahrscheinlich, daß zwei Dokumente denselben Fingerabdruck haben.
  2. Wenn ich den Fingerabdruck eines Dokuments habe, dann kann mir niemand ein anderes Dokument mit demselben Fingerabdruck unterschieben, da ich das durch Prüfung des Fingerabdrucks erkennen kann.

Ziel dieses Abschnittes ist, diese Anschauung etwas präziser zu machen und Anwendungen in der Kryptographie aufzuzeigen.

1. Klassische und kryptographische Hash-Funktionen

Wir lernen die Grundbegriffe kennen und erste Beispiele. Wichtig ist die Unterscheidung der (klassischen, normalen) Hash-funktionen von kryptographischen Hash-Funktionen.

1. Klassische und kryptographische Hash-Funktionen

2. Anwendungen von Hash-Funktionen

Motivation: Warum brauchen wir normale Hash-Funktionen?

Informelle Beschreibung: Eine Hash-Funktion ist eine Funktion $h\colon D \to K$, die ein langes Dokument $d \in D$ auf einen sehr kurzen Hash-Wert $h(d)$ abbildet.

Oft: $K = \{0,1\}^n$ mit relativ kleinem n, oft im Bereich 10–30.

Beispiel: Sei $d = \alpha_0\alpha_1\alpha_2\ldots\alpha_q$ wobei α_j Buchstaben in 8-bit Darstellung sind.
Wähle etwa: $h(\alpha_0\alpha_1\alpha_2\ldots\alpha_q) = \alpha_0 + \alpha_1 + \ldots + \alpha_q$ **mod** 1024.
Es könnte sein: $h(\text{Deutsches Grundgesetz}) = 314$

Hash Funktionen ergeben eine Art **Durchnummerierung** von Dokumenten.

Anwendung: Speicheradressen.

- Wir haben $2^{10} = 1024$ Speicher-Orte.
- Gegeben sei ein Dokument d.
- Lege das Dokument d in Speicherstelle $h(d)$ ab.

Beobachtung (1)

Beobachtung 1: Das geht nur für kleine Anzahl von Dokumenten.

- Bei größerer Zahl von Dokus: Risiko von Kollisionen: $h(d_1) = h(d_2)$ für $d_1 \neq d_2$.
- **Lösung 1:** Bei Dokumentenspeicherung Overflow-Bereiche einführen.
- **Lösung 2:** Mache die Länge der Hash-Werte ein klein wenig größer. Muß dann aber auch 2^k Speicherplätze vorhalten.

Frage: Was passiert im Extremfall, etwa bei $K = \{0,1\}^{256}$?

Wir haben nie so viele sinnvolle Dokumente

Kollisionsrisiko wird damit (fast) 0.

Fazit: k sollte für Fingerprints groß sein, Bereich 128-1024.

Beobachtung (2)

Beobachtung 2: Selbst bei großem k sind Kollisionen immer noch leicht zu finden.

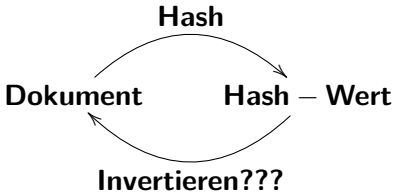
Vorhersagbare Änderungen

$h(\text{"Job begrüßte Alice ..."}) = h(\text{"Bob begrüßte Alice ..."}) + 8$
$h(\text{"Job begrüßte Alice ..."}) = h(\text{"Job begrüßte Adice ..."})$

Änderungen im Dokument schlagen **vorhersagbar** auf Hash-Werte durch.

Frage: Geht es auch anders, mit sehr schlechtem Vorhersagen?

Motivation für kryptographische Hash-Funktion



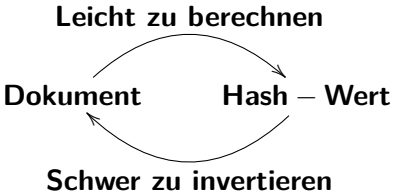


Abb. 1: Eine erste Motivation für Hash-Funktionen. Achtung: Schwer zu invertieren bedeutet hier genau genommen das Auffinden von Urbildern.

Umfrage: Präzisierung der Invertierbarkeit

Gegeben eine Funktion
f: Alle Dokumente der Länge 10000 $\rightarrow$ $\{0,1\}^{10}$

Kann diese Funktion invertierbar sein im Sinne der Existenz einer Umkehrfunktion?



https://twbk.de/21p1

Achtung: Konzept der Invertierbarkeit

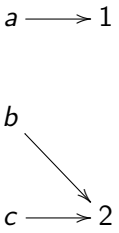


Abb. 2: Für eine Funktion $f : D \to H$, bei der H eine endliche Mengen ist und D eine echt größere Menge ist als H gibt es grundsätzlich **immer** Kollisionen $h(b) = h(c)$. Die Funktion ist daher **nicht invertierbar** im Sinne der Existenz inverser Funktionen. Man kann aber Urbilder suchen. Hier finden sich zwei Urbilder: $h(b) = 2$ und $h(c) = 2$

3 wichtige Eigenschaften kryptographischer Hash-Funktionen

Kollisionsresistent

Es ist schwierig, **zwei verschiedene Dokumente** d_1, d_2 zu finden,
die den **gleichen Hashwert** haben: $h(d_1) = h(d_2)$.

Schwer invertierbar (pre-image resistant)

Ist ein Wert w gegeben, dann ist es schwierig, ein **erstes Urbild** zu finden.

Das ist ein Dokument d mit dem Hash-Wert w zu finden: $h(d) = w$.

Schwer zweite Urbilder zu finden (second pre-image resistant)

Ist ein Dokument d_1 gegeben, dann ist es schwer, ein **zweites Urbild** zu finden.

Das ist ein weiteres Dokument d_2 mit dem selben Hash-Wert: $h(d_1) = h(d_2)$.

Existenz kryptographischer Hash-Funktionen

Problem 1: Wir müssen präzisieren, was "schwierig" genau bedeuten soll.
Das kann man (in der Komplexitätstheorie) tun.

Problem 2: Gibt es dann kryptographische Hash-Funktionen?
Das weiß man nicht.

Lösung 1: Man unterstellt die Existenz kryptographischer Hash-Funktionen.
Alle weitere Argumente basieren dann auf dieser Annahme.

Lösung 2: Man verwendet geeignete Funktionen.
Wird eine gebrochen, sucht man eine neue.

Typische Anforderung: Lawinen-Prinzip

Eine typische kryptographische Hash-Funktion ist eine Funktion

$$f : \text{Menge aller Dokumente} \to \{0,1\}^n$$

n liegt dabei im Bereich 128–1024.

Lawinen-Prinzip: Wenn man ein Bit im Input-Dokument ändert, dann

- ändert sich typischerweise rund die Hälfte aller Hashwert-Bits – und
- ist es sehr schwer, vorherzusagen, welche Bits sich ändern.

Typische Konstruktionsweise: Merkle-Damgard Konstruktion

Merkle-Damgard Konstruktion:

- Nehme an, Dokument bestehe aus k Blöcken der Länge n: $d = B_1 B_2 \dots B_k$
- Wähle Initialisierungsblock H_0
- Wähle Lawinen-artige Kompressionsfunktion $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$.
- Verkette: $H_1 = f(H_0 \cdot B_1)$, $H_2 = f(H_1 \cdot B_2)$, $H_3 = f(H_2 \cdot B_3)$
- Letzter Wert ist dann Hash-Wert des Dokuments.

Beachte: $\cdot$ bedeutet **hier** die Konkatenation (das Hintereinanderschreiben) von Strings.

Beispiele kryptographischer Hash-Funktionen

MD5:	128 Bit Wertebereich, gilt weithin als unsicher.
RIPEMD-128:	128 Bit, Kollisionen bekannt.
RIPEMD-320:	320 Bit, gilt noch als sicher.
SHA-0:	160 Bit Wertebereich, Kollisionen bekannt.
SHA-1:	160 Bit Wertebereich, Schwachstellen bekannt.
SHA-2:	Varianten mit 256 und 512 Bit und weitere.
SHA-3:	Varianten mit 256 und 512 Bit, neu seit 2015.
SHA-2	gilt als sicher gegen Angriffe durch Quanten-Computer, erlaubt aber Angriffe durch Längenerweiterung.
SHA-3	verhindert Angriffe durch Längenerweiterung, die Sicherheit gegen Quanten-Computer ist noch unbekannt.

Die **genauen** mathematischen Konstruktionen erscheinen konzeptuell zunächst einfach, sind im technischen Detail oft aber recht komplex ("Bit-Fummelei").

2. Anwendungen von Hash-Funktionen

Wo werden Hash-Funktionen nun angewendet? Und sind Angriffe gegen solche Anwendungen bekannt?

2. Anwendungen von Hash-Funktionen

Anwendung in der Integritäts-Sicherung einer Nachricht (1)

Situation: Integrität, aber nicht unbedingt Vertraulichkeit.

- Alice sendet eine Nachricht an Bob.
- **Schutzziel Integrität:** Angreifer soll Nachricht nicht unbemerkt verändern können.
- **Kein Schutzziel Vertraulichkeit:** Jeder darf die Nachricht lesen.

Beachte:

- Digitale Unterschrift würde das leisten, würde aber – unerwünscht – mehr leisten.
- Denn: Jeder könnte Alice das Schreiben der Nachricht nachweisen.

Lösung: Sogenannter MAC (**m**essage **a**uthentication **c**ode)

- Alice und Bob einigen sich auf eine pre-image resistente Hash-Funktion h.
- Alice und Bob vereinbaren ein geheimes Präfix p.
- Alice sendet das Dokument d und den Hash-Wert $m = h(p \cdot d)$.
- Bob erhält das Dokument d' und den Hash-Wert m'.
- Bob prüft, ob $h(p \cdot d') = m'$.

Anwendung in der Integritäts-Sicherung einer Nachricht (2)

Angriff:

- Mallory verändert d zu $\tilde{d}$.
- Wie kann sie nun m anpassen?
- **Gegeben** sind h, d, $h(p \cdot d)$ und $\tilde{d}$.
- **Gesucht** ist $h(p \cdot \tilde{d})$ (ohne Kenntnis von p).

Analyse:

- Da h pre-image resistent, ist es schwer, zu y ein x zu finden mit $y = h(x)$.
- Idee: Zu y und d ist es dann auch schwer, ein p zu finden mit $y = h(p \cdot d)$
  Dieses p braucht man, um $h(p \cdot \tilde{d})$ zu bestimmen.
  Also dürfte ein Angriff schwer sein?!

Achtung: Das stimmt, so formuliert, leider nicht.
Fehler 1: Pre-image Resistenz sagt streng genommen etwas anderes.
Fehler 2: Das p braucht man nicht unbedingt, um $h(p \cdot \tilde{d})$ zu bestimmen.

Längen-Erweiterungs-Angriff (1)

Viele etablierte Hash-Funktionen (SHA-1, SHA-2, MD-5) nutzen Merkle-Damgard.
Diese Konstruktion erlaubt es einem Angreifer

1. das Dokument d durch ein Suffix s zu erweitern: $d \mapsto \tilde{d} = d \cdot s$ und dann
2. den Hash-Wert $h(\tilde{d})$ des längeren Dokuments aus
   1. dem Suffix s und
   2. dem abgefangenen Hash-Wert $h(d)$ des kürzeren Dokuments

   zu berechnen.

Längen-Erweiterungs-Angriff (2)

Sei beispielsweise $d = B_1 \cdot B_2$.

Dann ist nach Merkle-Damgard $h(d) = f(f(H_0 \cdot B_1) \cdot B_2)$

Angreifer erweitert d zu $\tilde{d} = B_1 \cdot B_2 \cdot B$.

Angreifer kennt $h(d)$, also auch $f(f(H_0 \cdot B_1) \cdot B_2)$ (das ist $h(d)$).

Angreifer kennt aus den Standardisierungs-Dokumenten h und daher auch f.

Angreifer berechnet daraus $h(\tilde{d}) = f(\underbrace{f(f(H_0 \cdot B_1) \cdot B_2)}_{\text{Dem Angreifer bekannt}} \cdot B)$

Abwehr des Längen-Erweiterungs-Angriffs

Lösung 1: Bessere Hash-Funktion nutzen.

- Eine Hash-Funktion verwenden, die nicht nach Merkle-Damgard konstruiert ist.
- Beispiel: SHA-3.

Lösung 2: Besseren Mechanismus zur Berechnung des MAC nutzen.

- Beispiel: Der sog. HMAC mit verschachtelter Hash-Funktion: $h(k_1 \cdot h(k_2 \cdot x))$

Fazit: Kryptographie erfordert

- die Kenntnis des laufenden Fachdialogs – oder
- die Nutzung von Resultaten aus dem laufenden Fachdialog.

"Rant": Unternehmen, die "eigene" Kryptographie ohne tiefere Fachkenntnis anbieten, gibt es am Markt leider immer noch (sehr / zu) viele.

Weitere Anwendungen

Digitale Unterschriften:

- Unterschreibe Hash-Wert $h(d)$ des Dokuments und nicht das Dokument d selber.
- **Grund 1:** Blocklänge bei asymmetrischen Algorithmen eingeschränkt.
- **Grund 2:** Zusätzliche Sicherheit (etwa: gegen Homomorphie-Eigenschaften).

Kryptographische Bindung:

Alles, was bestätigt werden soll, in eine große Hash-Funktion "reinstecken".

Beispiel: Wenn Partner $h(\text{Kontonummer} \cdot \text{Betrag})$ kennt, dann kann Angreifer zwar nicht die Kontonummer oder den Betrag unbemerkt ändern, aber das Überweisungsdatum oder die Zahlungsreferenz.

Appendix

Übersicht

Übersicht 23

Verzeichnis aller Abbildungen Abb 24

Rechtliche Hinweise § 25

Zitierweise dieses Dokuments → 27

Verzeichnis aller Folien 28

Verzeichnis aller Abbildungen

1 Hash Funktionen 7
2 Hash Funktionen 9

Rechtliche Hinweise (1)

Die hier angebotenen Inhalte unterliegen deutschem Urheberrecht. Inhalte Dritter werden unter Nennung der Rechtsgrundlage ihrer Nutzung und der geltenden Lizenzbestimmungen hier angeführt. Auf das Literaturverzeichnis wird verwiesen. Das Zitatrecht in dem für wissenschaftliche Werke üblichen Ausmaß wird beansprucht. Wenn Sie eine Urheberrechtsverletzung erkennen, so bitten wir um Hinweis an den auf der Titelseite genannten Autor und werden entsprechende Inhalte sofort entfernen oder fehlende Rechtsnennungen nachholen. Bei Produkt- und Firmennamen können Markenrechte Dritter bestehen. Verweise und Verlinkungen wurden zum Zeitpunkt des Setzens der Verweise überprüft; sie dienen der Information des Lesers. Der Autor macht sich die Inhalte, auch in der Form, wie sie zum Zeitpunkt des Setzens des Verweises vorlagen, nicht zu eigen und kann diese nicht laufend auf Veränderungen überprüfen.

Alle sonstigen, hier nicht angeführten Inhalte unterliegen dem Copyright des Autors, Prof. Dr. Clemens Cap, ©2020. Wenn Sie diese Inhalte nützlich finden, können Sie darauf verlinken oder sie zitieren. Jede weitere Verbreitung, Speicherung, Vervielfältigung oder sonstige Verwertung außerhalb der Grenzen des Urheberrechts bedarf der schriftlichen Zustimmung des Rechteinhabers. Dieses dient der Sicherung der Aktualität der Inhalte und soll dem Autor auch die Einhaltung urheberrechtlicher Einschränkungen wie beispielsweise Par 60a UrhG ermöglichen.

Die Bereitstellung der Inhalte erfolgt hier zur persönlichen Information des Lesers. Eine Haftung für mittelbare oder unmittelbare Schäden wird im maximal rechtlich zulässigen Ausmaß ausgeschlossen, mit Ausnahme von Vorsatz und grober Fahrlässigkeit. Eine Garantie für den Fortbestand dieses Informationsangebots wird nicht gegeben.

Die Anfertigung einer persönlichen Sicherungskopie für die private, nicht gewerbliche und nicht öffentliche Nutzung ist zulässig, sofern sie nicht von einer offensichtlich rechtswidrig hergestellten oder zugänglich gemachten Vorlage stammt.

Use of Logos and Trademark Symbols: The logos and trademark symbols used here are the property of their respective owners. The YouTube logo is used according to brand request 2-9753000030769 granted on November 30, 2020. The GitHub logo is property of GitHub Inc. and is used in accordance to the GitHub logo usage conditions `https://github.com/logos` to link to a GitHub account. The Tweedback logo is property of Tweedback GmbH and here is used in accordance to a cooperation contract.

Rechtliche Hinweise (2)

Disclaimer: Die sich immer wieder ändernde Rechtslage für digitale Urheberrechte erzeugt ein nicht unerhebliches Risiko bei der Einbindung von Materialien, deren Status nicht oder nur mit unverhältnismäßig hohem Aufwand abzuklären ist. Ebenso kann den Rechteinhabern nicht auf sinnvolle oder einfache Weise ein Honorar zukommen, obwohl deren Leistungen genutzt werden.

Daher binde ich gelegentlich Inhalte nur als Link und nicht durch Framing ein. Lt EuGH Urteil 13.02.2014, C-466/12 (Pressemitteilung, Blog-Beitrag, Urteilstext). ist das unbedenklich, da die benutzten Links ohne Umgehung technischer Sperren auf im Internet frei verfügbare Inhalte verweisen.

Wenn Sie diese Rechtslage stört, dann setzen Sie sich für eine Modernisierung des völlig veralteten Vergütungs- und Anreizsystems für urheberrechtliche Leistungen ein. Bis dahin klicken Sie bitte auf die angegebenen Links und denken Sie darüber nach, warum wir keine für das digitale Zeitalter sinnvoll angepaßte Vergütungs- und Anreizsysteme digital erbrachter Leistungen haben.

Zu Risiken und Nebenwirkungen fragen Sie Ihren Rechtsanwalt oder Gesetzgeber.

Weitere Hinweise finden Sie im Netz hier und hier oder hier.

Zitierweise dieses Dokuments

Wenn Sie Inhalte aus diesem Werk nutzen oder darauf verweisen wollen,
zitieren Sie es bitte wie folgt:

Clemens H. Cap: Fingerprints und Hashes. Electronic document, 23. 11. 2026 https://iuk.one/DS-FINGER.pdf

Bibtex Information: https://iuk.one/DS-FINGER.bib

```
@misc{doc:DS-FINGER,
    author       = {Clemens H. Cap},
    title        = {Fingerprints und Hashes},
    year         = {2026},
    month        = {11},
    howpublished = {Electronic document},
    url          = {https://iuk.one/DS-FINGER.pdf }
}
```

Typographic Information:
Typeset on 2026-11-23 at 18:09
This is pdfTeX, Version 3.141592653-2.6-1.40.29 (TeX Live 2026) kpathsea version 6.4.2
This is pgf in version 3.1.11a
This is preamble-slides.tex ©C.H.Cap
Jobtype flag=foreground
Active modes (if any): slides, biblio
Keywords: keywords
Subject: subject

Verzeichnis aller Folien

Titelseite 1
Ziel 2

1. Klassische und kryptographische Hash-Funktionen 3

Motivation: Warum brauchen wir normale Hash-Funktionen? 4
Beobachtung (1) 5
Beobachtung (2) 6
Motivation für kryptographische Hash-Funktion 7
Umfrage: Präzisierung der Invertierbarkeit 8
Achtung: Konzept der Invertierbarkeit 9
3 wichtige Eigenschaften kryptographischer Hash-Funktionen 10
Existenz kryptographischer Hash-Funktionen 11
Typische Anforderung: Lawinen-Prinzip 12
Typische Konstruktionsweise: Merkle-Damgard Konstruktion 13
Beispiele kryptographischer Hash-Funktionen 14

2. Anwendungen von Hash-Funktionen 15

Anwendung in der Integritäts-Sicherung einer Nachricht (1) 16
Anwendung in der Integritäts-Sicherung einer Nachricht (2) 17
Längen-Erweiterungs-Angriff (1) 18
Längen-Erweiterungs-Angriff (2) 19
Abwehr des Längen-Erweiterungs-Angriffs 20
Weitere Anwendungen 21

Legende:
Fortsetzungsseite
Seite ohne Überschrift
Bildseite