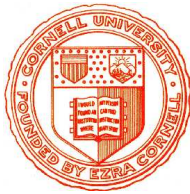


# Kryptographie und Komplexität



## Einheit 6

### Kryptographie und Sicherheit



1. Kryptographische Hashfunktionen
2. Passwörter und Identifikation
3. Digitale Signaturen
4. Secret Sharing
5. Anwendungen und Ausblick

- **Vertraulichkeit von Information**
  - Unbefugte sollen Nachrichten nicht lesen können
  - Haupteinsatzgebiet **kryptographischer Verschlüsselungsverfahren**

# SICHERHEIT HAT VIELE ASPEKTE

- **Vertraulichkeit von Information**
  - Unbefugte sollen Nachrichten nicht lesen können
  - Haupteinsatzgebiet **kryptographischer Verschlusselungsverfahren**
- **Datenintegrität**
  - Fälschung/Manipulation übermittelter Nachricht soll unmöglich sein
  - **Kryptographische Hashfunktionen** offenbaren nachträgliche Veränderung

# SICHERHEIT HAT VIELE ASPEKTE

- **Vertraulichkeit von Information**
  - Unbefugte sollen Nachrichten nicht lesen können
  - Haupteinsatzgebiet **kryptographischer Verschlüsselungsverfahren**
- **Datenintegrität**
  - Fälschung/Manipulation übermittelter Nachricht soll unmöglich sein
  - **Kryptographische Hashfunktionen** offenbaren nachträgliche Veränderung
- **Authentizität der Kommunikationsteilnehmer**
  - Identität eines Benutzers darf nicht von anderen angenommen werden
  - Möglich durch Verwendung von **Passwörtern und Identitätszertifikaten**

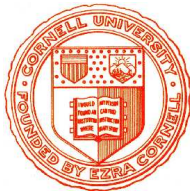
# SICHERHEIT HAT VIELE ASPEKTE

- **Vertraulichkeit von Information**
  - Unbefugte sollen Nachrichten nicht lesen können
  - Haupteinsatzgebiet **kryptographischer Verschlusselungsverfahren**
- **Datenintegrität**
  - Fälschung/Manipulation übermittelter Nachricht soll unmöglich sein
  - **Kryptographische Hashfunktionen** offenbaren nachträgliche Veränderung
- **Authentizität der Kommunikationsteilnehmer**
  - Identität eines Benutzers darf nicht von anderen angenommen werden
  - Möglich durch Verwendung von **Passwörtern und Identitätszertifikaten**
- **Verbindlichkeit digitaler Dokumente**
  - Absender darf Urheberschaft nicht nachträglich leugnen können
  - **Digitale Signaturen** binden Nachricht an ihren Absender

# SICHERHEIT HAT VIELE ASPEKTE

- **Vertraulichkeit von Information**
  - Unbefugte sollen Nachrichten nicht lesen können
  - Haupteinsatzgebiet **kryptographischer Verschlüsselungsverfahren**
- **Datenintegrität**
  - Fälschung/Manipulation übermittelter Nachricht soll unmöglich sein
  - **Kryptographische Hashfunktionen** offenbaren nachträgliche Veränderung
- **Authentizität der Kommunikationsteilnehmer**
  - Identität eines Benutzers darf nicht von anderen angenommen werden
  - Möglich durch Verwendung von **Passwörtern und Identitätszertifikaten**
- **Verbindlichkeit digitaler Dokumente**
  - Absender darf Urheberschaft nicht nachträglich leugnen können
  - **Digitale Signaturen** binden Nachricht an ihren Absender
- **Verwaltung von Hochsicherheitsgeheimnissen**
  - Geheimnis soll sicher gegen Verrat durch einzelne Berechtigte sein
  - **Secret Sharing** verteilt Geheimnis auf mehrere Personen

# Kryptographie und Komplexität



## Einheit 6.1

### Datenintegrität:

### Kryptographische Hashfunktionen



1. Kompressions- und Hashfunktionen
2. Konstruktion sicherer Hashfunktionen
3. Message Authentication Codes

- **Sicherheit gegenüber aktiven Angriffen**

- Angreifer könnte Nachrichten abfangen und modifiziert weiterleiten
- **Integrität**: Inhalt empfangener Nachrichten ist so wie verschickt
- **Authenizität**: Nachricht stammt wirklich vom angegebenen Absender



- **Sicherheit gegenüber aktiven Angriffen**
  - Angreifer könnte Nachrichten abfangen und modifiziert weiterleiten
  - **Integrität**: Inhalt empfangener Nachrichten ist so wie verschickt
  - **Authenizität**: Nachricht stammt wirklich vom angegebenen Absender
- **Verschlüsselung schützt nur gegen passive Angriffe**
  - Kennt der Angreifer die Art eines Textfragmentes (z.B. Zahl), so kann er die Nachricht durch Änderung einzelner Schlüsseltextbits manipulieren
  - Empfänger kann nicht feststellen, daß Nachricht angegriffen wurde (bestenfalls erkennt er, daß Inhalt nicht richtig sein kann)

- **Sicherheit gegenüber aktiven Angriffen**
  - Angreifer könnte Nachrichten abfangen und modifiziert weiterleiten
  - **Integrität**: Inhalt empfangener Nachrichten ist so wie verschickt
  - **Authenzität**: Nachricht stammt wirklich vom angegebenen Absender
- **Verschlüsselung schützt nur gegen passive Angriffe**
  - Kennt der Angreifer die Art eines Textfragmentes (z.B. Zahl), so kann er die Nachricht durch Änderung einzelner Schlüsseltextbits manipulieren
  - Empfänger kann nicht feststellen, daß Nachricht angegriffen wurde (bestenfalls erkennt er, daß Inhalt nicht richtig sein kann)
- **Schutz durch separate Kontrollinformation**
  - **Hashfunktionen** erzeugen “Checksum”, “Message Digest”, “Fingerprint”
  - Absender ergänzt Kontrollinformation zur Nachricht
  - Empfänger berechnet Hashwert aus Nachricht und vergleicht
  - Werte stimmen nur überein, wenn Nachricht unverändert

- **Hashfunktion**  $h:\Sigma^* \rightarrow \Sigma^n$ 
  - Abbildung beliebig langer Strings auf Strings fester Länge
  - z.B. Paritätsfunktion  $h(b_1\dots b_k) = b_1 \oplus b_2 \dots \oplus b_k$
  - Kryptographische Hashfunktionen müssen **effizient** berechenbar sein

# HASH- UND KOMPRESSIONSFUNKTIONEN

- **Hashfunktion**  $h:\Sigma^* \rightarrow \Sigma^n$ 
  - Abbildung beliebig langer Strings auf Strings fester Länge
  - z.B. Paritätsfunktion  $h(b_1\dots b_k) = b_1 \oplus b_2 \dots \oplus b_k$
  - Kryptographische Hashfunktionen müssen effizient berechenbar sein
- **Kompressionsfunktion**  $h:\Sigma^m \rightarrow \Sigma^n$ 
  - Abbildung von Datenblöcken auf Datenblöcke
  - z.B.  $h(b_1\dots b_7) = b_1 \oplus b_2 \dots \oplus b_7$  als Paritätswert in jedem Datenbyte
  - Hashfunktionen sind oft aus Kompressionsfunktionen zusammengesetzt
  - Kompressionsfunktionen werden oft als Hashfunktionen bezeichnet

- **Hashfunktion**  $h: \Sigma^* \rightarrow \Sigma^n$ 
  - Abbildung beliebig langer Strings auf Strings fester Länge
  - z.B. Paritätsfunktion  $h(b_1 \dots b_k) = b_1 \oplus b_2 \dots \oplus b_k$
  - Kryptographische Hashfunktionen müssen effizient berechenbar sein
- **Kompressionsfunktion**  $h: \Sigma^m \rightarrow \Sigma^n$ 
  - Abbildung von Datenblöcken auf Datenblöcke
  - z.B.  $h(b_1 \dots b_7) = b_1 \oplus b_2 \dots \oplus b_7$  als Paritätswert in jedem Datenbyte
  - Hashfunktionen sind oft aus Kompressionsfunktionen zusammengesetzt
  - Kompressionsfunktionen werden oft als Hashfunktionen bezeichnet
- **Sichere Hashfunktionen**
  - Angreifer darf nicht in der Lage sein, eine Nachricht zu manipulieren ohne daß sich ihr Hashwert ändert oder den Hashwert konsistent zu der Manipulation der Nachricht zu verändern

## ● Einwegfunktion

- Hashfunktion  $h$ , für die Urbilder nicht effizient zu bestimmen sind
- Für jedes  $x$  muß  $y = h(x)$  effizient zu berechnen sein
- Für fast alle  $y$  darf kein  $x$  mit  $y = h(x)$  effizient zu finden sein
- z.B.  $h_1(x) := x^e \bmod p$  oder  $h_2(x) := g^x \bmod p$  für ein  $g \in \mathbb{Z}_p$

## ● Einwegfunktion

- Hashfunktion  $h$ , für die Urbilder nicht effizient zu bestimmen sind
- Für jedes  $x$  muß  $y = h(x)$  effizient zu berechnen sein
- Für fast alle  $y$  darf kein  $x$  mit  $y = h(x)$  effizient zu finden sein
- z.B.  $h_1(x) := x^e \bmod p$  oder  $h_2(x) := g^x \bmod p$  für ein  $g \in \mathbb{Z}_p$

## ● Kollision

- Paar  $(x, x')$  mit  $x \neq x'$  aber  $h(x) = h(x')$
- Bei Byteparität z.B. 0011011 und 1001110

## ● Einwegfunktion

- Hashfunktion  $h$ , für die Urbilder nicht effizient zu bestimmen sind
- Für jedes  $x$  muß  $y = h(x)$  effizient zu berechnen sein
- Für fast alle  $y$  darf kein  $x$  mit  $y = h(x)$  effizient zu finden sein
- z.B.  $h_1(x) := x^e \bmod p$  oder  $h_2(x) := g^x \bmod p$  für ein  $g \in \mathbb{Z}_p$

## ● Kollision

- Paar  $(x, x')$  mit  $x \neq x'$  aber  $h(x) = h(x')$
- Bei Byteparität z.B. 0011011 und 1001110

## ● Schwache Kollisionsresistenz

- Für gegebenes  $x$  kann keine Kollision  $(x, x')$  effizient berechnet werden
- Gut für Prüfung der Übereinstimmung von Daten mit einem Original  
z.B. zeitliche Änderungen von Daten, Exaktheit von Kopien, etc.



## ● Einwegfunktion

- Hashfunktion  $h$ , für die Urbilder nicht effizient zu bestimmen sind
- Für jedes  $x$  muß  $y = h(x)$  effizient zu berechnen sein
- Für fast alle  $y$  darf kein  $x$  mit  $y = h(x)$  effizient zu finden sein
- z.B.  $h_1(x) := x^e \bmod p$  oder  $h_2(x) := g^x \bmod p$  für ein  $g \in \mathbb{Z}_p$

## ● Kollision

- Paar  $(x, x')$  mit  $x \neq x'$  aber  $h(x) = h(x')$
- Bei Byteparität z.B. 0011011 und 1001110

## ● Schwache Kollisionsresistenz

- Für gegebenes  $x$  kann keine Kollision  $(x, x')$  effizient berechnet werden
- Gut für Prüfung der Übereinstimmung von Daten mit einem Original  
z.B. zeitliche Änderungen von Daten, Exaktheit von Kopien, etc.

## ● Starke Kollisionsresistenz

- Es ist nicht möglich, irgendeine Kollision  $(x, x')$  effizient zu bestimmen
- Schützt digitale Unterschriftensysteme gegen “beliebige” Fälschungen

- **Jede stark kollisionsresistente Hashfunktion  $h$  ist auch schwach kollisionsresistent**
  - Wenn  $h$  nicht schwach kollisionsresistent wäre, dann könnte man eine beliebige Kollision  $(x, x')$  effizient bestimmen, indem man ein festes  $x$  wählt und dazu eine Kollision berechnet

- **Jede stark kollisionsresistente Hashfunktion  $h$  ist auch schwach kollisionsresistent**
  - Wenn  $h$  nicht schwach kollisionsresistent wäre, dann könnte man eine beliebige Kollision  $(x, x')$  effizient bestimmen, indem man ein festes  $x$  wählt und dazu eine Kollision berechnet
- **Jede nicht injektive stark kollisionsresistente Hashfunktion  $h$  ist auch eine Einwegfunktion**
  - Wenn  $h$  keine Einwegfunktion wäre, dann könnte man eine beliebige Kollision  $(x, x')$  effizient bestimmen, indem man ein festes  $x$  wählt,  $h(x)$  berechnet und dazu ein Urbild  $x'$  berechnet
  - $x'$  ist mit Wahrscheinlichkeit größer als  $1/2$  verschieden von  $x$ , wenn der Definitionsbereich  $\Sigma^m$  von  $h$  mindestens doppelt so groß wie der Bildbereich  $\Sigma^n$  von  $h$  ist

## VERGLEICH DER SICHERHEITSKRITERIEN

- **Jede stark kollisionsresistente Hashfunktion  $h$  ist auch schwach kollisionsresistent**
  - Wenn  $h$  nicht schwach kollisionsresistent wäre, dann könnte man eine beliebige Kollision  $(x, x')$  effizient bestimmen, indem man ein festes  $x$  wählt und dazu eine Kollision berechnet
- **Jede nicht injektive stark kollisionsresistente Hashfunktion  $h$  ist auch eine Einwegfunktion**
  - Wenn  $h$  keine Einwegfunktion wäre, dann könnte man eine beliebige Kollision  $(x, x')$  effizient bestimmen, indem man ein festes  $x$  wählt,  $h(x)$  berechnet und dazu ein Urbild  $x'$  berechnet
  - $x'$  ist mit Wahrscheinlichkeit größer als  $1/2$  verschieden von  $x$ , wenn der Definitionsbereich  $\Sigma^m$  von  $h$  mindestens doppelt so groß wie der Bildbereich  $\Sigma^n$  von  $h$  ist

Stark kollisionsresistente injektive Hashfunktionen sind nicht notwendigerweise Einwegfunktionen

# BERECHNUNGS-AUFWAND FÜR ATTACKEN AUF $h:\Sigma^m \rightarrow \Sigma^n$

**Kollisionen sind leichter zu finden als Urbilder**

## Kollisionen sind leichter zu finden als Urbilder

### ● Brute-Force Attacke auf Einwegfunktionen

- Um zu einem gegebenen  $y$  mit Wahrscheinlichkeit  $\epsilon \geq 1/2$  ein Urbild  $x$  zu finden, muß ein Angreifer  $\frac{|\Sigma|^n}{2}$  zufällige Kandidaten aus  $\Sigma^m$  prüfen

## Kollisionen sind leichter zu finden als Urbilder

### ● Brute-Force Attacke auf Einwegfunktionen

- Um zu einem gegebenen  $y$  mit Wahrscheinlichkeit  $\epsilon \geq 1/2$  ein Urbild  $x$  zu finden, muß ein Angreifer  $\frac{|\Sigma|^n}{2}$  zufällige Kandidaten aus  $\Sigma^m$  prüfen

### ● Geburtstagsattacke auf starke Kollisionsresistenz

- Um zu mit Wahrscheinlichkeit  $\epsilon \geq 1/2$  eine Kollision  $(x, x')$  zu finden, muß ein Angreifer  $1.2 \cdot |\Sigma|^{n/2}$  Kandidatenpaare prüfen

- Anzahl ergibt sich aus Geburtstagsparadox (siehe §2.3):

Die Chance eine Kollision von Hashwerten zu finden entspricht der Chance eine Kollision von Geburtstagen zu finden

## Kollisionen sind leichter zu finden als Urbilder

### ● Brute-Force Attacke auf Einwegfunktionen

- Um zu einem gegebenen  $y$  mit Wahrscheinlichkeit  $\epsilon \geq 1/2$  ein Urbild  $x$  zu finden, muß ein Angreifer  $\frac{|\Sigma|^n}{2}$  zufällige Kandidaten aus  $\Sigma^m$  prüfen

### ● Geburtstagsattacke auf starke Kollisionsresistenz

- Um zu mit Wahrscheinlichkeit  $\epsilon \geq 1/2$  eine Kollision  $(x, x')$  zu finden, muß ein Angreifer  $1.2 \cdot |\Sigma|^{n/2}$  Kandidatenpaare prüfen

- Anzahl ergibt sich aus Geburtstagsparadox (siehe §2.3):

Die Chance eine Kollision von Hashwerten zu finden entspricht der Chance eine Kollision von Geburtstagen zu finden

Starke Kollisionsresistenz liefert mehr Sicherheit als Einwegfunktionen

- Das leichtere Problem (Kollisionen finden) wird “unlösbar” gemacht
- Um Geburtstagsattacken zu verhindern, muß Blockgröße  $n \geq 128$  sein



- **Gibt es echte Kollisionsresistenz?**

- $\mathcal{P} \neq \mathcal{NP}$  ist eine notwendige Voraussetzung

- Zeige, daß Berechnung von  $h$  in  $\mathcal{P}$  liegt

- Zeige, daß Finden von Urbildern oder Kollisionen  $\mathcal{NP}$ -vollständig ist

## ● Gibt es echte Kollisionsresistenz?

- $\mathcal{P} \neq \mathcal{NP}$  ist eine notwendige Voraussetzung
  - Zeige, daß Berechnung von  $h$  in  $\mathcal{P}$  liegt
  - Zeige, daß Finden von Urbildern oder Kollisionen  $\mathcal{NP}$ -vollständig ist
- $\mathcal{NP}$ -Vollständigkeit alleine reicht nicht
  - Komplexitätsklassen beruhen auf Worst-Case Analysen
  - Sicherheitsbeweise verlangen Average-Case oder Best-Case Analysen

## ● Gibt es echte Kollisionsresistenz?

- $\mathcal{P} \neq \mathcal{NP}$  ist eine notwendige Voraussetzung
  - Zeige, daß Berechnung von  $h$  in  $\mathcal{P}$  liegt
  - Zeige, daß Finden von Urbildern oder Kollisionen  $\mathcal{NP}$ -vollständig ist
- $\mathcal{NP}$ -Vollständigkeit alleine reicht nicht
  - Komplexitätsklassen beruhen auf Worst-Case Analysen
  - Sicherheitsbeweise verlangen Average-Case oder Best-Case Analysen
- Praktische Anwendungen müssen Hashfunktionen verwenden, deren Kollisionsresistenz bisher nicht widerlegt ist

## ● Gibt es echte Kollisionsresistenz?

- $\mathcal{P} \neq \mathcal{NP}$  ist eine notwendige Voraussetzung
  - Zeige, daß Berechnung von  $h$  in  $\mathcal{P}$  liegt
  - Zeige, daß Finden von Urbildern oder Kollisionen  $\mathcal{NP}$ -vollständig ist
- $\mathcal{NP}$ -Vollständigkeit alleine reicht nicht
  - Komplexitätsklassen beruhen auf Worst-Case Analysen
  - Sicherheitsbeweise verlangen Average-Case oder Best-Case Analysen
- Praktische Anwendungen müssen Hashfunktionen verwenden, deren Kollisionsresistenz bisher nicht widerlegt ist

## ● Verwende kryptographische Funktionen

- Einfache Möglichkeit **parametrisierte Hashfunktionen** zu erzeugen
- Konstruiere  $h: \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  aus  $e_K: \{0, 1\}^n \rightarrow \{0, 1\}^n$ ,  $K \in \{0, 1\}^n$   
z.B.  $h(K, x) = e_K(x) \oplus x$        $h(K, x) = e_K(x) \oplus x \oplus K$   
 $h(K, x) = e_K(x \oplus K) \oplus x$      $h(K, x) = e_K(x \oplus K) \oplus x \oplus K \dots$
- Hashfunktion scheint sicher wenn Verschlüsselungsfunktion sicher ist
- Nur Verschlüsselungsfunktionen mit Blockgröße  $n \geq 128$  geeignet

Iterierte Kompressionsfunktion  $g: \{0, 1\}^m \rightarrow \{0, 1\}^n$

Iterierte Kompressionsfunktion  $g: \{0, 1\}^m \rightarrow \{0, 1\}^n$

- **Verzicht auf Parameter in Kompressionsfunktion**
  - Parameter müssten als geheime Schlüssel zuvor ausgetauscht werden
  - Verwende Initialwert/Nachrichtenteile/Zwischenhash stattdessen

## Iterierte Kompressionsfunktion $g: \{0, 1\}^m \rightarrow \{0, 1\}^n$

- **Verzicht auf Parameter in Kompressionsfunktion**
  - Parameter müssten als geheime Schlüssel zuvor ausgetauscht werden
  - Verwende Initialwert/Nachrichtenteile/Zwischenhash stattdessen
- **Vorbereitung: Padding von  $x \in \{0, 1\}^*$** 
  - Ergänze führende Nullen, bis Länge Vielfaches von  $r = m - n$  ist
  - Ergänze Kontrolldaten am Ende, um Kollisionsresistenz zu erhalten
    - Ein Block von  $r$  Nullen zur Abgrenzung
    - $|x|$  als Binärzahl, ergänzt um Nullen bis Länge Vielfaches von  $r - 1$  und weiter ergänzt um Einsen an jeder  $r - 1$ -ten Stelle

## Iterierte Kompressionsfunktion $g: \{0, 1\}^m \rightarrow \{0, 1\}^n$

- **Verzicht auf Parameter in Kompressionsfunktion**
  - Parameter müssten als geheime Schlüssel zuvor ausgetauscht werden
  - Verwende Initialwert/Nachrichtenteile/Zwischenhash stattdessen
- **Vorbereitung: Padding von  $x \in \{0, 1\}^*$** 
  - Ergänze führende Nullen, bis Länge Vielfaches von  $r = m - n$  ist
  - Ergänze Kontrolldaten am Ende, um Kollisionsresistenz zu erhalten
    - Ein Block von  $r$  Nullen zur Abgrenzung
    - $|x|$  als Binärzahl, ergänzt um Nullen bis Länge Vielfaches von  $r - 1$  und weiter ergänzt um Einsen an jeder  $r - 1$ -ten Stelle
- **Iteration der Kompression**
  - Zerlege Gesamtstring in Wörter  $x_1 x_2 \dots x_k$  der Länge  $r$
  - Wähle Initialwert  $H_0 \in \{0, 1\}^n$  (z.B.  $H_0 := 0^n$ )
  - In Schritt  $i \leq k$  berechne  $H_i = g(H_{i-1} \circ x_i)$
  - Ergebnis ist  $h(x) = H_k$



- **$h$  ist kollisionsresistent wenn dies für  $g$  gilt**

Sei  $(x, x')$  eine Kollision von  $h$  mit zugehörigen Folgen  $x_1 \dots x_k, x'_1 \dots x'_{k'}$  und  $H_0 \dots H_k, H'_1 \dots H'_{k'}$ . Wir konstruieren eine Kollision für  $g$ .

- **$h$  ist kollisionsresistent wenn dies für  $g$  gilt**

Sei  $(x, x')$  eine Kollision von  $h$  mit zugehörigen Folgen  $x_1 \dots x_k, x'_1 \dots x'_{k'}$  und  $H_0 \dots H_k, H'_1 \dots H'_{k'}$ . Wir konstruieren eine Kollision für  $g$ .

– Wegen  $h(x) = h(x')$  muß  $H_k = H'_{k'}$  gelten

– Gilt  $H_{k-j-1} \neq H'_{k'-j-1}$  aber  $H_{k-j} = H'_{k'-j}$  für ein  $j \leq t$  ★

dann ist  $(H_{k-j-1} \circ x_{k-j}, H'_{k'-j-1} \circ x'_{k'-j})$  eine Kollision von  $g$

## ● $h$ ist kollisionsresistent wenn dies für $g$ gilt

Sei  $(x, x')$  eine Kollision von  $h$  mit zugehörigen Folgen  $x_1 \dots x_k, x'_1 \dots x'_{k'}$  und  $H_0 \dots H_k, H'_1 \dots H'_{k'}$ . Wir konstruieren eine Kollision für  $g$ .

– Wegen  $h(x) = h(x')$  muß  $H_k = H'_{k'}$  gelten

– Gilt  $H_{k-j-1} \neq H'_{k'-j-1}$  aber  $H_{k-j} = H'_{k'-j}$  für ein  $j \leq t$  ★

dann ist  $(H_{k-j-1} \circ x_{k-j}, H'_{k'-j-1} \circ x'_{k'-j})$  eine Kollision von  $g$

– Wenn ★ nicht gilt, dann muß  $H_{k-j} = H'_{k'-j}$  für alle  $j \leq t$  gelten

Andererseits muß wegen  $x \neq x'$  auch  $x_{k-i} \neq x'_{k'-i}$  für ein  $i$  gelten

Es folgt  $H_{k-i-1} \circ x_{k-i} \neq H'_{k'-i-1} \circ x'_{k'-i}$  und somit

$$H_{k-i} = g(H_{k-i-1} \circ x_{k-i}) = H'_{k'-i} = g(H'_{k'-i-1} \circ x'_{k'-i})$$

Damit ist  $(H_{k-i-1} \circ x_{k-i}, H'_{k'-i-1} \circ x'_{k'-i})$  eine Kollision von  $g$

## ● $h$ ist kollisionsresistent wenn dies für $g$ gilt

Sei  $(x, x')$  eine Kollision von  $h$  mit zugehörigen Folgen  $x_1 \dots x_k, x'_1 \dots x'_{k'}$  und  $H_0 \dots H_k, H'_1 \dots H'_{k'}$ . Wir konstruieren eine Kollision für  $g$ .

– Wegen  $h(x) = h(x')$  muß  $H_k = H'_{k'}$  gelten

– Gilt  $H_{k-j-1} \neq H'_{k'-j-1}$  aber  $H_{k-j} = H'_{k'-j}$  für ein  $j \leq t$  ★

dann ist  $(H_{k-j-1} \circ x_{k-j}, H'_{k'-j-1} \circ x'_{k'-j})$  eine Kollision von  $g$

– Wenn ★ nicht gilt, dann muß  $H_{k-j} = H'_{k'-j}$  für alle  $j \leq t$  gelten

Andererseits muß wegen  $x \neq x'$  auch  $x_{k-i} \neq x'_{k'-i}$  für ein  $i$  gelten

Es folgt  $H_{k-i-1} \circ x_{k-i} \neq H'_{k'-i-1} \circ x'_{k'-i}$  und somit

$$H_{k-i} = g(H_{k-i-1} \circ x_{k-i}) = H'_{k'-i} = g(H'_{k'-i-1} \circ x'_{k'-i})$$

Damit ist  $(H_{k-i-1} \circ x_{k-i}, H'_{k'-i-1} \circ x'_{k'-i})$  eine Kollision von  $g$

– Also ist die Bestimmung von Kollisionen für  $g$  genauso leicht wie für  $h$

- **Integration effizienter Binäroperationen**
  - $\oplus$  Verknüpfungen mit speziellen Strings, Rotationen, Shifts
  - Effizienter als Einbau kryptographischer Funktionen

- **Integration effizienter Binäroperationen**
  - $\oplus$  Verknüpfungen mit speziellen Strings, Rotationen, Shifts
  - Effizienter als Einbau kryptographischer Funktionen
- **Message-Digest Familie (MD4/MD5)**
  - **MD4**: 3 Runden mit je 16 Schritten, zu brechen mit  $2^{20}$  Aufrufen
  - **MD5**: Verstärkte Version von MD4 mit 128-Bit Kompression  
Nur angreifbar, wenn Angreifer Initialwerte festlegen darf

- **Integration effizienter Binäroperationen**
  - $\oplus$  Verknüpfungen mit speziellen Strings, Rotationen, Shifts
  - Effizienter als Einbau kryptographischer Funktionen
- **Message-Digest Familie (MD4/MD5)**
  - **MD4**: 3 Runden mit je 16 Schritten, zu brechen mit  $2^{20}$  Aufrufen
  - **MD5**: Verstärkte Version von MD4 mit 128-Bit Kompression  
Nur angreifbar, wenn Angreifer Initialwerte festlegen darf
- **Secure Hash Familie (SHA-1, SHA-256)**
  - **SHA-1**: 4 Runden mit je 20 komplexen Schritten, 160-Bit Kompression  
meist verwendeter Algorithmus in der Praxis, sehr effizient
  - Das NIST empfiehlt ab 2010 die Verwendung der Variante **SHA-256**

- **Integration effizienter Binäroperationen**
  - $\oplus$  Verknüpfungen mit speziellen Strings, Rotationen, Shifts
  - Effizienter als Einbau kryptographischer Funktionen
- **Message-Digest Familie (MD4/MD5)**
  - **MD4**: 3 Runden mit je 16 Schritten, zu brechen mit  $2^{20}$  Aufrufen
  - **MD5**: Verstärkte Version von MD4 mit 128-Bit Kompression  
Nur angreifbar, wenn Angreifer Initialwerte festlegen darf
- **Secure Hash Familie (SHA-1, SHA-256)**
  - **SHA-1**: 4 Runden mit je 20 komplexen Schritten, 160-Bit Kompression  
meist verwendeter Algorithmus in der Praxis, sehr effizient
  - Das NIST empfiehlt ab 2010 die Verwendung der Variante **SHA-256**
- **RIPEMD Familie (RIPEMD-128, RIPEMD-160)**
  - **RIPEMD-160**: 5 Runden mit je 16 parallel ausführbaren Schritten  
160-Bit Kompression, schnellster akzeptierter Algorithmus
  - 128-Bit Variante (**RIPEMD-128**) wurde erfolgreich angegriffen



# MESSAGE AUTHENTICATION CODES

## Prüfung von Integrität und Identität

## Prüfung von Integrität und Identität

- **Koppelung von Schlüssel und Fingerprint**
  - MAC wird aus Nachricht und geheimem Schlüssel bestimmt
  - Nur Sender und Empfänger kennen den Schlüssel  $K$
  - MAC ist kurzer Datensatz, Länge unabhängig von Nachrichtengröße

## Prüfung von Integrität und Identität

- **Koppelung von Schlüssel und Fingerprint**
  - MAC wird aus Nachricht und geheimem Schlüssel bestimmt
  - Nur Sender und Empfänger kennen den Schlüssel  $K$
  - MAC ist kurzer Datensatz, Länge unabhängig von Nachrichtengröße
- **Hilfsmittel: Parametrisierte Hashfunktion**
  - Familie von Hashfunktionen  $h_K: \Sigma^* \rightarrow \Sigma^n$  für Schlüssel  $K \in \mathcal{K}$

## Prüfung von Integrität und Identität

- **Koppelung von Schlüssel und Fingerprint**
  - MAC wird aus Nachricht und geheimem Schlüssel bestimmt
  - Nur Sender und Empfänger kennen den Schlüssel  $K$
  - MAC ist kurzer Datensatz, Länge unabhängig von Nachrichtengröße
- **Hilfsmittel: Parametrisierte Hashfunktion**
  - Familie von Hashfunktionen  $h_K: \Sigma^* \rightarrow \Sigma^n$  für Schlüssel  $K \in \mathcal{K}$
- **Authentifizierungsverfahren**
  - Sender und Empfänger tauschen geheimen Schlüssel  $K$  aus
  - Sender berechnet  $MAC = h_K(x)$  für Nachricht  $x$  und schickt  $x \circ MAC$
  - Empfänger empfängt  $x' \circ MAC'$  und vergleicht  $MAC'$  mit  $h_K(x')$
  - Im Erfolgsfall ist Nachricht unverfälscht und Absender authentisch
  - Wichtige Annahme: Angreifer kann ohne Kenntnis von  $K$  zu keiner Nachricht einen gültigen MAC bestimmen

# ANGRIFFE AUF MESSAGE AUTHENTICATION CODES

- **No message attack**

- Angreifer kennt Hashfamilie aber nicht den konkreten Schlüssel
- Angreifer versucht ein gültiges Paar  $x \circ MAC$  zu bestimmen

# ANGRIFFE AUF MESSAGE AUTHENTICATION CODES

- **No message attack**

- Angreifer kennt Hashfamilie aber nicht den konkreten Schlüssel
- Angreifer versucht ein gültiges Paar  $x \circ MAC$  zu bestimmen

- **Known message attack**

- Angreifer kennt gültige Paare  $(x_1, MAC_1), \dots, (x_q, MAC_q)$
- Angreifer versucht ein neues gültiges Paar zu erzeugen
- **Replay Attacke**: Angreifer schickt eines der abgefangenen Paare
- Nur mit zusätzlichen Techniken erkennbar

# ANGRIFFE AUF MESSAGE AUTHENTICATION CODES

- **No message attack**

- Angreifer kennt Hashfamilie aber nicht den konkreten Schlüssel
- Angreifer versucht ein gültiges Paar  $x \circ MAC$  zu bestimmen

- **Known message attack**

- Angreifer kennt gültige Paare  $(x_1, MAC_1), \dots, (x_q, MAC_q)$
- Angreifer versucht ein neues gültiges Paar zu erzeugen
- **Replay Attacke**: Angreifer schickt eines der abgefangenen Paare
- Nur mit zusätzlichen Techniken erkennbar

- **Chosen message attack**

- Angreifer kann MACs für selbstgewählte  $x_1, \dots, x_q$  bestimmen ohne  $K$
- **$(\epsilon, q)$  forgery**: Angreifer versucht mit Wahrscheinlichkeit  $\epsilon > 0$  ein neues gültiges Paar zu erzeugen
- Ein MAC ist **sicher**, wenn keine chosen message attack effizient ist

## ● CBC-MAC: Iterative Konstruktion

- Verwende Kryptosystem mit  $e_K: \{0, 1\}^m \rightarrow \{0, 1\}^m$
- Zerlege Bitstring (nach Padding) in Wörter  $x_1x_2\dots x_t$  der Länge  $m$
- Wähle Initialwert  $y_0 \in \{0, 1\}^m$  (z.B.  $y_0 := 0^m$ )
- In Schritt  $i \leq t$  berechne  $y_i = e_K(y_{i-1} \oplus x_i)$
- Ergebnis ist  $MAC(K, x) = y_t$



## ● CBC-MAC: Iterative Konstruktion

- Verwende Kryptosystem mit  $e_K: \{0, 1\}^m \rightarrow \{0, 1\}^m$
- Zerlege Bitstring (nach Padding) in Wörter  $x_1x_2\dots x_t$  der Länge  $m$
- Wähle Initialwert  $y_0 \in \{0, 1\}^m$  (z.B.  $y_0 := 0^m$ )
- In Schritt  $i \leq t$  berechne  $y_i = e_K(y_{i-1} \oplus x_i)$
- Ergebnis ist  $MAC(K, x) = y_t$

## ● XOR-MAC: Parallele Bearbeitung

- Zerlege Bitstring in Wörter  $x_1x_2\dots x_t$  der Länge  $m - \lceil \log t \rceil - 1$
- Wähle Initialwert  $IV \in \{0, 1\}^{m-1}$  und berechne  $y_0 = e_K(0 \circ IV)$
- Für alle  $i \leq t$  berechne  $y_i = e_K(1 \circ i \circ x_i)$  (Zahl  $i$  in Binärdarstellung)
- Ergebnis ist  $MAC(K, x) = y_0 \oplus y_1 \oplus \dots \oplus y_t$

## ● CBC-MAC: Iterative Konstruktion

- Verwende Kryptosystem mit  $e_K: \{0, 1\}^m \rightarrow \{0, 1\}^m$
- Zerlege Bitstring (nach Padding) in Wörter  $x_1x_2\dots x_t$  der Länge  $m$
- Wähle Initialwert  $y_0 \in \{0, 1\}^m$  (z.B.  $y_0 := 0^m$ )
- In Schritt  $i \leq t$  berechne  $y_i = e_K(y_{i-1} \oplus x_i)$
- Ergebnis ist  $MAC(K, x) = y_t$

## ● XOR-MAC: Parallele Bearbeitung

- Zerlege Bitstring in Wörter  $x_1x_2\dots x_t$  der Länge  $m - \lceil \log t \rceil - 1$
- Wähle Initialwert  $IV \in \{0, 1\}^{m-1}$  und berechne  $y_0 = e_K(0 \circ IV)$
- Für alle  $i \leq t$  berechne  $y_i = e_K(1 \circ i \circ x_i)$  (Zahl  $i$  in Binärdarstellung)
- Ergebnis ist  $MAC(K, x) = y_0 \oplus y_1 \oplus \dots \oplus y_t$
- Chiffrierung von Initialwert und Blocknummer erhöht Sicherheit bei gleichartigen Teilblöcken

## ● CBC-MAC: Iterative Konstruktion

- Verwende Kryptosystem mit  $e_K: \{0, 1\}^m \rightarrow \{0, 1\}^m$
- Zerlege Bitstring (nach Padding) in Wörter  $x_1x_2\dots x_t$  der Länge  $m$
- Wähle Initialwert  $y_0 \in \{0, 1\}^m$  (z.B.  $y_0 := 0^m$ )
- In Schritt  $i \leq t$  berechne  $y_i = e_K(y_{i-1} \oplus x_i)$
- Ergebnis ist  $MAC(K, x) = y_t$

## ● XOR-MAC: Parallele Bearbeitung

- Zerlege Bitstring in Wörter  $x_1x_2\dots x_t$  der Länge  $m - \lceil \log t \rceil - 1$
- Wähle Initialwert  $IV \in \{0, 1\}^{m-1}$  und berechne  $y_0 = e_K(0 \circ IV)$
- Für alle  $i \leq t$  berechne  $y_i = e_K(1 \circ i \circ x_i)$  (Zahl  $i$  in Binärdarstellung)
- Ergebnis ist  $MAC(K, x) = y_0 \oplus y_1 \oplus \dots \oplus y_t$
- Chiffrierung von Initialwert und Blocknummer erhöht Sicherheit bei gleichartigen Teilblöcken

## ● Sicherheit

- Bester bekannter Angriff (Geburtstagsattacke) liefert  $(\frac{1}{2}, 2^{m/2})$  forgery

## ● Einbau von Verschlüsselung in Hashfunktion

- Verwende parameterfreie Hashfunktion  $h:\{0, 1\}^* \rightarrow \{0, 1\}^n$
- Konstruiere  $h_K(x) = h(K \oplus opad \circ h(K \oplus opad \circ x))$   
wobei  $K \in \mathcal{K}$  und  $ipad, opad$  verschiedene Konstanten
- Addition der Konstanten erzeugt aus  $K$  zwei verschiedene Schlüssel zur Erhöhung der Sicherheit

## ● Einbau von Verschlüsselung in Hashfunktion

- Verwende parameterfreie Hashfunktion  $h:\{0,1\}^* \rightarrow \{0,1\}^n$
- Konstruiere  $h_K(x) = h(K \oplus opad \circ h(K \oplus opad \circ x))$   
wobei  $K \in \mathcal{K}$  und  $ipad, opad$  verschiedene Konstanten
- Addition der Konstanten erzeugt aus  $K$  zwei verschiedene Schlüssel zur Erhöhung der Sicherheit

## ● Verwendung in der Praxis

- Konstruktion des MAC mit MD5 oder SHA-1 und 512 Bit Schlüsseln sowie festen Byte-Konstanten  $ipad = 3636...36$ ,  $opad = 5C5C...5C$

## ● Einbau von Verschlüsselung in Hashfunktion

- Verwende parameterfreie Hashfunktion  $h:\{0, 1\}^* \rightarrow \{0, 1\}^n$
- Konstruiere  $h_K(x) = h(K \oplus opad \circ h(K \oplus opad \circ x))$   
wobei  $K \in \mathcal{K}$  und  $ipad, opad$  verschiedene Konstanten
- Addition der Konstanten erzeugt aus  $K$  zwei verschiedene Schlüssel zur Erhöhung der Sicherheit

## ● Verwendung in der Praxis

- Konstruktion des MAC mit MD5 oder SHA-1 und 512 Bit Schlüsseln sowie festen Byte-Konstanten  $ipad = 3636...36$ ,  $opad = 5C5C...5C$
- Einsatz beim Aufbau sicherer Kanäle
  - (1) Berechne den MAC der verschlüsselten Nachricht IPSEC
  - (2) Berechne den MAC vor Verschlüsselung der Nachricht SSH
  - (3) Verschlüssele Klartext und MAC SSL
- (2) und (3) sind (im Prinzip) angreifbar, (1) ist nachweislich sicher