Die Klausur

Dienstag den 19.2. um 9 Uhr im Felix-Klein-Hörsaal

Für die Klausur ...

... sind Definitionen ganz zentral.

Eine Aufgabe könnte sein:

a) Wofür ist "PPT" in "PPT-Algorithmus" die Abkürzung?

b) Es sei eine Folge $(a_n)_{n \in \mathbf{N}}$ gegeben. Unter welcher Bedingung wird die Folge vernachlässigbar genannt?

c) Sind die folgenden Folgen vernachlässigbar (ohne Begründung)?

$$\frac{1}{n^5} \quad , \quad \frac{1}{3^n} \quad , \quad \frac{n^{23}}{2^n} \quad , \quad \frac{1}{n^n}$$

Aktueller Überblick

§0 Einführende Worte (✓)

§1 Geschichtlicher Überblick (✓)

§2 Zufall (✓)

§3 Perfekte Sicherheit und ihre Grenzen (✓)

§4 Angriffsszenarien (✓)

§5 Der komplexitätstheoretische Ansatz(✓)

§6 Pseudozufallsgeneratoren und Stromchiffren (✓)

§7 Pseudozufallsfunktionen und Blockchiffren (✓)

§8 Message Authentication Codes und Hash-Funktionen

§9 Kryptographie mit öffentlichen Schlüsseln

§7 Message Authentication Codes und Hash-Funktionen

(IND-CCA2-sichere Verschlüsselungssysteme)

Wiederholung: Sicherheitsniveaus

Das Ununterscheidbarkeits-Spiel zum Lauscher-Angriff

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $k := \mathcal{G}en(1^n)$.
3. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{0, 1\}$ uniform. Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{0, 1\}$.

Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Ununterscheidbarkeits-Spiel zum Lauscher-Angriff

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $k := \mathcal{G}en(1^n)$.
3. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{0,1\}$ uniform. Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{0,1\}$.

Der Angreifer hat gewonnen, wenn $i = j$ ist.

Wir definieren den Vorteil oder Erfolg von $\mathcal{A}$ als

$$2 \cdot (\mathbf{P}[\mathcal{A} \text{ gewinnt}] - \frac{1}{2}) \, .$$

Das Spiel zum CPA

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $k := \mathcal{G}en(1^n)$.
3. Der Angreifer kann beliebige Nachrichten m mit dem Schlüssel k verschlüsseln lassen (d.h. $\mathcal{E}nc_k(m)$ berechnen lassen).
4. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
5. Der Herausforderer wählt $i \in \{0, 1\}$ uniform. Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer kann beliebige (!) Nachrichten mit dem Schlüssel k verschlüsseln lassen.
7. Der Angreifer entscheidet sich für $j \in \{0, 1\}$.

Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Spiel zum nicht-adaptiven CCA

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $k := \mathcal{G}en(1^n)$.
3. Der Angreifer kann beliebige Texte ver- und entschlüsseln lassen.
4. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
5. Der Herausforderer wählt $i \in \{0, 1\}$ uniform. Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer kann beliebige Nachrichten verschlüsseln lassen.
7. Der Angreifer entscheidet sich für $j \in \{0, 1\}$.

Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Spiel zum adaptiven CCA (CCA2)

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $k := \mathcal{G}en(1^n)$.
3. Der Angreifer kann beliebige Texte ver- und entschlüsseln lassen.
4. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
5. Der Herausforderer wählt $i \in \{0, 1\}$ uniform. Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer kann beliebige Nachrichten verschlüsseln lassen und Chiffriertexte verschieden von c entschlüsseln lassen.
7. Der Angreifer entscheidet sich für $j \in \{0, 1\}$.

Der Angreifer hat gewonnen, wenn $i = j$ ist.

Was erfüllt was?

Gegeben ein Bit-Generator,

Verschlüsselung mit XOR (synchrone Stromchiffre)

Wenn der Generator ein Pseudozufallsgenerator ist,

ist die Chiffre sicher gegenüber Lauschern (IND-EAV-sicher).

Gegeben eine Blockchiffre

Verschlüsselung mit OFB

Wenn die Blockchiffre auf eine Pseudozufallsfunktion beschreibt,

ist die Chiffre sicher gegenüber Angriffen mit gewähltem Klartext (IND-CPA-sicher).

Und weiter?

Es sei eine eine Blockchiffre gegeben,

diese beschreibt eine Funktionenfamilie $(f_{k_0})_{k_0}$.

Wir haben auch die "Umkehrungs-Funktionenfamilie" $(f_{k_0}^{-1})_{k_0}$.

Optimistische Sicherheitsannahme: Wir haben eine starke Pseudozufallspermutation.

Und weiter?

Betrachte "OFB-Modus mit einem Block":

$\mathcal{E}nc$. Eingabe: k, m

Wähle $r \in \{0,1\}^n$ uniform.

Ausgabe $c := (r, f_k(r) \oplus m)$.

$\mathcal{D}ec$. Eingabe $c = (r, a)$

Ausgabe $f_k(r) \oplus a$ $(= m,$ falls $c = \mathcal{E}nc_k(m)$.

Dies ist nicht CCA2-sicher:

Im CCA2-Spiel: Gegeben $c = (r, a)$, frage nach Entschlüsselung von (r, a') mit $a' \neq a$.

Und weiter?

Betrachte “CBC-Modus mit einem Block”:

$\mathcal{E}nc$. Eingabe: k, m

Wähle $r \in \{0,1\}^n$ uniform.

Ausgabe $(r, f_k(r \oplus m))$.

$\mathcal{D}ec$. Eingabe $c = (r, a)$

Ausgabe $f_k^{-1}(a) \oplus r$ $(= m,$ falls $c = \mathcal{E}nc_k(m))$.

Dies ist nicht CCA2-sicher:

Im CCA2-Spiel: Gegeben $c = (r, a)$, frage nach Entschlüsselung von (r', a) mit $r' \neq r$.

Message Authentication Codes

Schwäche und Lösungsansatz

Schwäche

Ein Angreifer kann dem Herausforderer beliebige Chiffriertexte unterjubeln. Diese werden alle entschlüsselt.

Lösungsansatz

Nachrichten werden “unterschrieben”, nur Nachrichten mit gültiger Unterschrift werden entschlüsselt.

Weiterer Vorteil

Angreifer können keine gefälschten Nachrichten einspielen.

Der Begriff MAC

Ein Message Authentication Code (MAC) (Nachrichten-Authentisierungscode) ist eine Art private Signaturen.

Merkwürdig: "Message Authentication Code" ("MAC") steht im Englischen eher für die Verfahren.

Und dann heißt der Code ("tag").

Definition

Ein MAC-Verfahren oder MAC besteht aus:

- Einem Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.

  Eingabe: 1^n mit $n =$ Sicherheitsparameter

  Ausgabe: Schlüssel $k = \mathcal{G}en(1^n)$
- Einem Markierungs-Erzeugungsalgorithmus $\mathcal{M}ac$
  Eingabe: Schlüssel k,
  Nachricht $m \in \{0,1\}^*$ oder $m \in \{0,1\}^\ell \quad (\ell = \ell(n))$

  Ausgabe: Markierung $t = \mathcal{M}ac_k(m)$
- Einem deterministischen Verifikationsalgorithmus $\mathcal{V}rfy$.
  Eingabe: Schlüssel k, Nachricht m, Markierung t

  Ausgabe: $\mathcal{V}rfy_k(m,t) \in \{0 = \text{“ungültig”}, 1 = \text{“gültig”}\}$

mit: $\mathcal{V}rfy_k^{(0)}(m^{(0)}, \mathcal{M}ac_{k^{(0)}}(m^{(0)})) = 1$ (mit Wahrscheinlichkeit 1)
für alle relevanten $k^{(0)}, m^{(0)}$.

Kanonische Verifikation

Wenn $\mathcal{M}ac$ determinstisch ist, kann man so verifizieren:

Gegeben (m, t), teste ob $t = \mathcal{M}ac_k(m)$.

Spiel für die Sicherheit von MACs

Gegeben: $\mathcal{G}en, \mathcal{M}ac, \mathcal{V}rft$.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $k := \mathcal{G}en(1^n)$.
3. Der Angreifer kann beliebige Markierungen erzeugen lassen, d.h. für Nachricht m nach $\mathcal{M}ac_k(m)$ fragen.
4. Der Angreifer gibt ein Tupel (m, t) aus.

Zwei Möglichkeiten für "gewonnen":

1. Normale Variante. Der Angreifer hat gewonnen, wenn dies gültig ist ($\mathcal{V}rfy_k(m, t) = 1$) und er nicht nach m gefragt hat.
2. Starke Variante. Der Angreifer hat gewonnen, wenn dies gültig ist ($\mathcal{V}rfy_k(m, t) = 1$) und das Tupel (m, t) nicht bei "Frage / Antwort" vorgekommen ist.

Sicherheit von MACs

Definition

Ein Message Authentication Code ist sicher, wenn jeder PPT-Angreifer bezüglich der normalen Variante eine vernachlässigbare Gewinnwahrscheinlichkeit hat.

Ein Message Authentication Code ist stark sicher, wenn jeder PPT-Angreifer bezüglich der starken Variante eine vernachlässigbare Gewinnwahrscheinlichkeit hat.

Bemerkung

Wenn der Markierungs-Erzeugungsalgorithmus deterministisch ist und das Verfahren sicher ist, dann ist es auch stark sicher.

MACs basierend auf Pseudozufallsfunktionen

MACs mit fester Länge

Es sei $(f_{k^{(0)}})_{k^{(0)}}$ eine Funktionenfamilie.

Wie immer sei n der Sicherheitsparameter.

Konstruktion

... eines MACs für Nachrichten der Länge n.

$\mathcal{M}ac$.

Eingabe: k, m.

Ausgabe: $t := f_k(m)$

$\mathcal{V}rfy$.

kanonisch.

Satz. Wenn die Funktionsfamilie eine starke Pseudozufallsfunktion ist, ist der MAC sicher für Nachrichten der festen Länge n.

CBC-MAC

Vom CBC-Modus inspiriertes MAC für längere Nachrichten.

Zuerst für feste Blockanzahl $\ell(n)$.

Algorithmus für MAC

Eingabe. Schlüssel k, Nachricht m

1. Schreibe $m = m_1 \dots m_\ell$ mit $|m_i| = n$
2. Setze $t := 0^n$
3. Für $i = 1, \dots z$:
   Setze $t := f_k(t \oplus m_i)$
4. Gib t aus.

Verifikation kanonisch (für Nachrichten der festen Länge).

Satz. Wenn die Funktionsfamilie eine Pseudozufallsfunktion ist, dann ist der MAC sicher für Nachrichten der Länge $\ell(n) \cdot n$.

Bemerkung. "Feste Länge" ist wirklich wichtig. Es werden hier nur Nachrichten der vorgegebenen Länge betrachtet.

Unsicherheit bei beliebiger Blocklänge

Angreifer

1. Wähle m_1, m_1' verschieden mit Länge n / 1 Block.
2. Frage nach
   $t_1 := \mathcal{M}ac_k(m_1) = f_k(m_1)$
   $t_1' := \mathcal{M}ac_k(m_1') = f_k(m_1')$.
3. Wähle m_2 mit Länge n / 1 Block.
4. Frage nach $t := \mathcal{M}ac_k(m_1 m_2)$

   [Es ist

   $$t = f_k(t_1 \oplus m_2) = f_k(t_1' \oplus (t_1 \oplus t_1' \oplus m_2)) = \mathcal{M}ac_k(m_1' m_2') \; .$$

   mit $m_2' := t_1 \oplus t_1' \oplus m_2$.]
5. Gib $(m_1' m_2', t)$ aus.

CBC-MAC

Für beliebig lange Blocklängen (Nachrichten mit Länge $x \cdot n$):

Zwei Möglichkeiten:

1. Füge Länge $|m|$ zu Beginn der Nachricht an (nicht am Ende!).
2. Wähle zwei Schlüssel k_1, k_2.
   Bereche t mit k_1 wie zuvor.
   Gib $f_{k_2}(t)$ aus.

Satz. Wenn die Funktionsfamilie eine Pseudozufallsfunktion ist, dann sind beide Konstruktionen eines MAC sicher.

CBC-MAC

Frage. Warum sollte die Länge nicht am Ende angehängt werden?

Authentisierte Verschlüsselung

Unfälschbarkeit

Neben CCA2-Sicherheit haben wir noch ein Ziel:

Der Empfänger soll gefälschte Nachrichten erkennen können.

Wir betrachten wieder ein Spiel.

Wie immer haben wir einen beliebigen Angreifer $\mathcal{A}$.

Spiel zur Unfälschbarkeit

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $k := \mathcal{G}en(1^n)$.
3. Der Angreifer kann beliebige Nachrichten verschlüsseln lassen.
4. Der Angreifer gibt einen String (angeblichen Chiffriertext) c aus.

Es sei $m := \mathcal{D}ec_k(c)$.

Der Angreifer hat gewonnen, wenn $\mathcal{E}nc_k(m) = c$ und er nicht nach $\mathcal{E}nc_k(m)$ gefragt hat.

Der Erfolg von $\mathcal{A}$ ist hier definiert als $\mathbf{P}[\mathcal{A}$ gewinnt$]$.

Variante. Der Angreifer kann auch Nachrichten entschlüsseln lassen.

Unfälschbarkeit

Definition. Ein Verschlüsselungsverfahren ist unfälschbar, wenn jeder PPT-Angreifer im Unfälschbarkeitsspiel nur vernachlässigbaren Erfolg hat.

Definition. Ein Verschlüsselungsverfahren ist authentisiert, wenn es sowohl CCA2-sicher als auch unfälschbar ist.

Konstruktionen für authentisierte Verschlüsselung

Wie sollte man MACs einsetzen?

Es ist naheliegend, zwei Schlüssel k_E, k_M zu benutzen.

Mögliche Varianten

Gegeben: k_E, k_M, m.

1. Verschlüsseln und authentisieren.
   Berechne
   $$c := \mathcal{E}nc_{k_E}(m), t := \mathcal{M}ac_{k_M}(m) \ ,$$
   sende (c, t).

   Der Empfänger entschlüsselt zu einer Nachricht m. Er verifiziert die markierte Nachricht (m, t).

Konstruktionen für authentisierte Verschlüsselung

2. Zuerst authentisieren, dann verschlüsseln.
   Berechne

$$t := \mathcal{M}ac_{k_M}(m) \quad , \quad c := \mathcal{E}nc_{k_E}(m|t)$$

   Sende c.
   Der Empfänger entschlüsselt zuerst. Wenn das Resultat als $m|t$ interpretierbar ist, verifiziert er diese markierte Nachricht.

3. Zuerst verschlüsseln, dann authentisieren.
   Berechne

$$c := \mathcal{E}nc_{k_E}(m) \quad , \quad t := \mathcal{M}ac_{k_M}(c)$$

   Sende (c, t).
   Der Empfänger verifiziert zuerst die markierte Nachricht.
   Wenn dies OK ist, entschlüsselt er.

Konstruktion für authentisierte Verschlüsselung

Wir wollen so ein Resultat:

Wenn das Verschlüsselungsverfahren CPA-sicher ist und der Nachrichten-Authentisierungscode sicher ist, dann erhält man mit der Konstruktion ein authentisierte Verschlüsselungsverfahren.

Modularisierung!

Konstruktion für authentisierte Verschlüsselung

Verschlüsseln und authentisieren.

Berechne

$$c := \mathcal{E}nc_{k_E}(m) \quad , \quad t := \mathcal{M}ac_{k_M}(m) \, ,$$

sende (c, t).

Der Empfänger entschlüsselt zu einer Nachricht m. Er verifiziert die markierte Nachricht (m, t).

Kann total unsicher sein, weil ein MAC nichts mit Verschlüsselung zu tun haben muss.

Konstruktionen für authentisierte Verschlüsselung

Zuerst authentisieren, dann verschlüsseln.

Berechne

$$t := \mathcal{M}ac_{k_M}(m) \quad , \quad c := \mathcal{E}nc_{k_E}(m|t)$$

Sende c.

Der Empfänger entschlüsselt zuerst. Wenn das Resultat als $m|t$ interpretierbar ist, verifiziert er diese markierte Nachricht.

Problem. Fehlermeldungen

Fehlermeldungen

Allgemein

Bei einem Verschlüsselungsverfahren fordern wir

$$\mathcal{D}ec_k(\mathcal{E}nc_k(m)) = 1$$

für alle k, m.

Aber was, wenn $\mathcal{D}ec$ auf ein Paar (k, c) angewandt wird, für das es gar keine Nachricht gibt?

Wichtiger Spezialfall

Eine Blockchiffre wird in einem Modus angewandt.

Die Länge der Nachricht muss passend gemacht werden (padding).

Padding-Fehler

PKCS #5-padding

Sei m eine Nachricht in Bytes.

An die Nachricht werden Bytes angehängt. Jedes dieser Bytes kodiert die Fülllänge ($\rightarrow$ kodierte Daten)

Wenn eine Nachricht gerade so in Blöcke passt, wird ein ganzer Block angehängt.

Nach dem Entschlüsseln der kodierten Daten wird das Padding überprüft. Wenn's nicht passt, wird ein padding-Fehler ausgegeben.

Wenn nun CBC-Mode angewandt wird:

Ohne Authentisierung ist das total unsicher.

Konstruktionen für authentisierte Verschlüsselung

Zuerst authentisieren, dann verschlüsseln.

Berechne

$$t := \mathcal{M}ac_{k_M}(m) \quad , \quad c := \mathcal{E}nc_{k_E}(m|t)$$

Sende c.

Der Empfänger entschlüsselt zuerst. Wenn das Resultat als $m|t$ interpretierbar ist, verifiziert er diese markierte Nachricht.

Problem. Fehlermeldungen

Angenommen:

Padding wie beschrieben wird benutzt. Es gibt eine entsprechende Fehlermeldung.

Diese Fehlermeldung ist erkennbar (verschieden von einem Verifikations-Fehler).

Dann ist das Verfahren genau unsicher wie ohne MAC.

Konstruktionen für authentisierte Verschlüsselung

Zuerst verschlüsseln, dann authentisieren.

Berechne

$$c := \mathcal{E}nc_{k_E}(m) \quad , \quad t := \mathcal{M}ac_{k_M}(c)$$

Sende (c, t).

Der Empfänger verifiziert zuerst die markierte Nachricht. Wenn dies OK ist, entschlüsselt er.

Satz. Die Konstruktion "verschlüsseln-dann-authentisieren" ergibt ein authentisiertes Verschlüsselungsverfahren, wenn das ursprüngliche Verschlüsselungsverfahren IND-CPA-sicher ist, der Nachrichten-Authentisierungscode stark sicher ist und die beiden Schlüssel k_E, k_M unabhängig voneinander "nach Vorschrift" gewählt werden.

Hash-Funktionen

Definition von Hash-Funktionen

Eine Hash-Funktion ist erstmal eine (in Polynomzeit) berechenbare Funktion $H : \{0,1\}^* \longrightarrow \{0,1\}^n$.

Wir wollen, dass so eine Hash-Funktion H kollisionsresistent sind.

D.h. es soll keine zwei Strings x, x' mit $H(x) = H(x')$ "geben".

Dafür müssen wir mindestens n als Sicherheitsparameter zulassen.

Das geht auch nicht so gut / reicht nicht.

Lösung. Man betrachtet Hash-Funktionen mit Schlüssel.

Definition von Hash-Funktionen

Man kann nun eine Hash-Funktion als ein Paar von Algorithmen $\mathcal{G}en, \mathcal{H}$ definieren.

Man kann dann unterscheiden zwischen Hash-Funktionen mit fester Eingabelänge und Hash-Funktionen mit variabler Eingabelänge.

Wenn $\mathcal{G}en$ als kanonisch angenommen wird, kann man es auch weglassen.

Dann kann man auch sagen:

Eine Hash-Funktion ist eine Funktionsfamilie $(H_{k^{(0)}})_{k^{(0)} \in \{0,1\}^*}$ mit:

Für $n := |k^{(0)}|$ ist $H_{k^{(0)}} : ... \longrightarrow \{0,1\}^{\ell(n)}$

und $H_{k^{(0)}}(s^{(0)})$ ist in Polynomzeit berechenbar.

Definition von Hash-Funktionen

Spiel

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $k := \mathcal{G}en(1^n)$ und sendet ihn dem Angreifer (!).
3. Der Angreifer gibt zwei Strings $x \neq x'$ aus.

Der Angreifer hat gewonnen, wenn $\mathcal{H}_k(x) = \mathcal{H}_k(x')$ ist. ("Er hat eine Kollision gefunden.")

Definition von Hash-Funktionen

Definition. Eine Hash-Funktion ist kollisionsresistent, wenn kein PPT-Angreifer mit nicht-vernachlässigbarer Wahrscheinlichkeit eine Kollision findet.

Die Merkle-Damgard-Transformation

Wir wollen aus einer Hash-Funktion h für feste Eingabelänge eine Hash-Funktion H mit viel längerer Eingabelänge machen.

Es sei h eine Hash-Funktion mit Eingabelänge $2n$ und Ausgabelänge n.

Konstruktion von H

Eingabe: Schlüssel k, String x mit $|x| < 2^n$.

1. Hänge 0-en an x an, bis Länge ein Vielfaches von n ist.
   Sei $B := \frac{|x|}{n}$ die Anzahl der Blöcke.
   Seien $x_1, \ldots, x_B$ die Blöcke, sei $x_{B+1} := |x|$ (im 2-er System).
2. Setze $z := 0$.
3. Für $i = 1, \ldots B + 1$:
   Setze $z := h_k(z|x_i)$
4. Gib z aus.

Die Merkle-Damgard-Transformation

Satz. Wenn h eine kollisionsresistente Hash-Funktion ist, dann ist die Hash-Funktion H beruhend auf der Merkle-Damgard-Transformation auch kollisionsresistent.

Anwendung. MD5

Hash-Funktionen und MACs

Hash-and-MAC

Naheliegend:

Ein MAC = Nachrichten-Authentisierungscode(-Verfahren) = privates Unterschriftsverfahren kann auch “auf einen Hash-Wert” angewandt werden.

Wenn die Hash-Funktion kollisionsresistent ist, sollte das genauso sicher sein.

Das ist auch so.

Hash-and-MAC

Es sei ein MAC $(\mathcal{G}en, \mathcal{M}ac, \mathcal{V}rfy)$ und eine Hash-Funktion H gegeben.

Konstruktion

Neue Schlüsselerzeugung. Wähle k_M, k_H wie vorgeschrieben unabhängig von einander.

Neuer Markierungs-Algorithmus. Gegeben m, gib $\mathcal{M}ac_{k_M}(H_{k_H}(m))$ aus.

Neuer Verifikations-Algorithmus. Gegeben (m, t), überprüfe, ob $\mathcal{V}rfy_{k_M}(H_{k_H}(m)) = 1$.

Hash-and-MAC

Satz. Für ein sicheres MAC und eine kollisionsresistente Hash-Funktion ergibt die Konstruktion Hash-and-MAC ein sicheres MAC.

HMAC

Frage. Kann man auch mit einer Hash-Funktion ein sicheres MAC erhalten?

Z.B. so?

Gegeben Hash-Funktion-Schlüssel k_H, MAC-Schlüssel k_M und Nachricht m, gib $H_{k_H}(k_M|m)$ aus.

Antwort. Nein mit der Merkle-Damgard-Transformation (wenn der Schlüssel k_H dem Angreifer bekannt ist).

[Das ist ein praxisrelevanter Angriff für MD5.]

HMAC

Es sei h eine Hash-Funktion mit Eingabelänge $2n$ und Ausgabelänge n.

Es sei H die Hash-Funktion erhalten durch die Merke-Damgard-Transformation.

Wir konstrukieren einen MAC mit Schlüssel $k := (k_h, k_i, k_o)$ mit $|k_i| = |k_o| = n$.

Gegeben k, m, berechne $H_{k_h}(k_o | H_{k_h}(k_i | m))$.

Sollte sicher sein ...

HMAC in der Praxis

Die Hash-Funktion hat keinen Schlüssel.

$k_o = k_i \oplus$ (fester String) (bzw. padding)