Message Authentication Codes

Entspricht Hashfunktionen mit geheimen Schlüsseln.

$h : K \times M \rightarrow H$, $MAC = h_k(m)$.

- h parametrisierte Hashfunktion.
- m Nachricht.
- k geheimer Schlüssel.

Mit der Nachricht m wird $h_k(m)$ übertragen. Der Empfänger berechnet $h_k(m)$ aus m und k und vergleicht mit dem gesendeten MAC.

Liefert Datenintegrität und Authentizität.

Sicherheitsmodell

Der ideale MAC ist wieder eine Zufallsfunktion, für jedes $k \in K$ soll h_k nicht von einer „zufällig" gewählten Funktion $M \rightarrow H$ unterscheidbar sein.

Spezieller betrachtet man folgendes Sicherheitsmodell:

- Ein Angreifer darf $h_k(m_i)$ für beliebig von ihm vorgegebene Nachrichten m_i erhalten.
- Der Angreifer gewinnt (erzeugt eine Fälschung), wenn er ein m und y mit $y = h_k(m)$ berechnet.
- MAC ist sicher, wenn es keinen effizienten Angreifer gibt, der mit signifikanter Wahrscheinlichkeit gewinnt.

Konstruktion von MACs

Folgende Konstruktionstypen können unterschieden werden:

- MACs aus Blockchiffren im CBC Modus.
- MACs aus MDCs.
- Spezielle Konstruktionen.

CBC-MACs sind sehr weit verbreitet.

- Standardisiert z.B. in FIPS-113 von 1985.

CBC-MAC

Verwendet eine Blockchiffre $E : K \times \{0,1\}^b \rightarrow \{0,1\}^b$.

CBC-MAC von Nachricht $M \in \{0,1\}^*$ unter Schlüssel k:

- Schreibe $M = M_1 || \ldots || M_n$ mit $M_i \in \{0,1\}^b$ (und Padding).
- $y_0 \leftarrow IV \leftarrow 0^b$.
- Für $i \leftarrow 1, \ldots, n$:
  $y_i \leftarrow \mathcal{E}(k, y_{i-1} \oplus m_i)$.
- Ausgabe y_n.

In anderen Worten: Der CBC-MAC ist der letzte Chiffretextblock der Verschlüsselung von m mit $\mathcal{E}$ im CBC Modus, unter Verwendung des konstanten $IV = 0^b$.

Vorteil: Leicht aus bestehenden Teilen programmiert.

CBC-MAC Padding und Postprocessing

In den Standards sind drei Padding Varianten vorgesehen:

- Nullen anhängen.
- Eine Eins und Nullen anhängen.
- Nullen anhängen und zusätzlichen Block mit Nachrichtenlänge.

Man kann ein optionales Postprocessing vornehmen:

- Wähle Schlüssel k_1. Dann MAC-Wert $\mathcal{E}(k, \mathcal{D}(k_1, y_n))$. Entspricht EDE-Verschlüsselung im letzten Schritt.
- Wähle Schlüssel k_1. Dann MAC-Wert $\mathcal{E}(k_1, y_n)$. Entspricht EE-Verschlüsselung (nicht besonders gut).

Erschwert exhaustive Key-search.

CBC-MAC Sicherheit

Gilt als sicher,

- wenn Blockchiffre sicher ist und
- wenn die Nachrichtenlänge konstant ist.

Man kann zeigen: Ist die Blockchiffre eine pseudozufällige Funktion, so auch der CBC-MAC. Nur durch mindestens ungefähr $2^{b/2}$ Anfragen an die Blockchiffre kann der CBC-MAC von einer zufälligen Funktion unterschieden werden.

Ist aber unsicher, wenn die Nachrichtenlänge nicht konstant ist:

- Erfrage $y_1 \leftarrow h_k(m_1)$.
- Erfrage $y_2 \leftarrow h_k(y_1||m_2)$.
- Nun gilt $y_2 = h_k(m_1||0^b||m_2)$. Liefert eine Fälschung.

Abhilfe: $h_{h'(|m|)}(m)$ oder $h_k(|m|\,||m)$ verwenden.

CBC-MAC Sicherheit

Geburtstagsangriff auf CBC-MAC mit fester Nachrichtenlänge d:

- Wähle $1.18 \cdot 2^{b/2}$ Nachrichten $m_i = m_{1,i}||m_{2,i}||m_3$ mit $m_{1,i} \in \{0,1\}^b$ paarweise verschieden, $m_{2,i} \in \{0,1\}^b$ zufällig und $m_3 \in \{0,1\}^{d-2b}$ beliebig.
- Erfrage alle $h_k(m_i)$. Es ergibt sich eine Kollision $h_k(m_i) = h_k(m_j)$ für $i \neq j$ mit Wahrscheinlichkeit $\geq 1/2$.
- Nun gilt $h_k(m_{1,i}) \oplus m_{2,i} = h_k(m_{1,j}) \oplus m_{2,j}$.
- Wähle beliebiges $m_\delta \in \{0,1\}^b$ und erfrage $y \leftarrow h_k(m_{1,i}||(m_{2,i} \oplus m_\delta)||m_3)$.
- Nun gilt $y = h_k(m_{1,j}||(m_{2,j} \oplus m_\delta)||m_3)$. Liefert eine Fälschung.

Folgerung: $2^{b/2}$ bestmögliche Sicherheit bem CBC-MAC.

MACs aus MDCs

Gegeben eine Hashfunktion $h : \{0,1\}^* \to \{0,1\}^b$.

Drei einfache Varianten:

1. MAC $= h(k||m)$
2. MAC $= h(m||k)$
3. MAC $= h(k_1||m||k_2)$

Liefern für ideale Hashfunktion idealen MAC. Für iterierte Hashfunktionen aber nicht besonders sicher.

Annahme: h iteriert, also $h_0 = IV$, $h_{i+1} = f(h_i, m_i)$, ...

zu 1. Aus $h(k||m)$ kann leicht $h(k||m||m')$ bzw. $h(k||m||p||m')$ ausgerechnet werden, wobei p das Padding der Hashfunktion ist.

MACs aus MDCs

zu 2. Finde Kollision $h(m) = h(m')$ („offline“ möglich). Erfrage $y \leftarrow h(m||k)$. Dann ist $y = h(m'||k)$ eine Fälschung.
Ist im Prinzip ein Hash-then-encrypt Ansatz.

zu 3. Durch Erfragen der MACs findet man eine Kollision $h(k_1||m_i||k_2) = h(k_1||m_j||k_2)$. Dann gilt auch $h(k_1||m_i) = h(k_1||m_j)$.
Man sucht nun einfach in K nach k_1 und dann nach k_2.
Der Aufwand ist damit $2\#K$ statt $\#K^2$ ...

Die Konstruktionen 1-3 werden daher so nicht verwendet.

Geschachtelte MACs

Seien $g : K_1 \times \{0,1\}^* \to \{0,1\}^m$ und $h : K_2 \times \{0,1\}^m \to \{0,1\}^n$ mit $m \geq n$.

Wir werden zeigen: Ist g_{k_1} kollisionsresistent bei unbekanntem Schlüssel und h_{k_2} ein sicherer MAC, so ist $h_{k_2} \circ g_{k_1}$ ein sicherer MAC.

Wir betrachten dazu folgende Angreifer:

1. Kollisionsangriff bei unbekanntem Schlüssel: k_1 ist geheim, der Angreifer erhält trotzdem die Werte $g_{k_1}(m)$ für m seiner Wahl. Er versucht eine Kollision $g_{k_1}(m_i) = g_{k_1}(m_j)$ mit $m_i \neq m_j$ zu finden.
2. Kleiner MAC Angreifer: Angreifer gegen h_{k_2}.
3. Großer MAC Angreifer: Angreifer gegen $h_{k_2} \circ g_{k_1}$.

Ein (ε, q)-Angreifer für 1-3 führt einen erfolgreichen Angriff bei zufälliger Schlüsselwahl mit Wahrscheinlichkeit $\geq \varepsilon$ und $\leq q$ Orakelanfragen aus.

Geschachtelte MACs

Thm: k_1 und k_2 seien unabhängig und zufällig. Es gebe keinen $(\varepsilon_1, q+1)$-Angreifer gegen 1 und keinen (ε_2, q)-Angreifer gegen 2. Für jeden (ε, q)-Angreifer gegen 3 gilt dann $\varepsilon \leq \varepsilon_1 + \varepsilon_2$.

Bew: Sei A_3 ein (ε, q)-Angreifer gegen 3. Seien (m_i, z_i) für $1 \leq i \leq q$ die Orakelanfragen von A_3 und (m, z) die Ausgabe von A_3. Wir definieren $y_i = g_{k_1}(m_i)$ und $y = g_{k_1}(m)$. Mit Wahrscheinlichkeit $\geq \varepsilon$ ist (m, z) eine Fälschung, und wir nehmen dies nun an.

Der Angreifer A_1 gegen 1 wird wie folgt definiert: Er wählt ein zufälliges k_2 und beantwortet die Orakelanfragen von A_3 mit seinem Orakel für g_{k_1} und $h_{k_2}(y_i)$. Durch Orakelanfrage berechnet er $y = g_{k_1}(m)$ und gibt gegebenenfalls eine Kollision $y = y_i$ aus. Dies sind $q+1$ Anfragen an das Orakel. Außerdem gilt $m \neq m_i$.

Geschachtelte MACs

Der Angreifer A_2 gegen 2 wird wie folgt definiert: Er wählt ein zufälliges k_1 und beantwortet die Orakelanfragen m_i von A_3 mit $g_{k_1}(m_i)$ und seinem Orakel für h_{k_2}. Er gibt $(g_{k_1}(m), z)$ als korrektes Ergebnis aus, falls $g_{k_1}(m) \neq g_{k_1}(m_i)$ für alle i. Das Orakel h_{k_2} wurde dann wie erforderlich nicht nach $g_{k_1}(m)$ gefragt.

A_1 hat nach Annahme Erfolgswahrscheinlichkeit $\leq \varepsilon_1$.
A_2 hat nach Annahme Erfolgswahrscheinlichkeit $\leq \varepsilon_2$.
A_1 und A_2 rufen A_3 auf.
Für A_3 macht es keinen Unterschied, ob er von A_1 oder A_2 aufgerufen wird.

Geschachtelte MACs

Zufallsquelle von A_3 als Eingabebitstring $\sigma \in \{0,1\}^s$ auffassen, A_1 und A_2 unter Einbeziehung der Orakel zu deterministischen Algorithmen A_1', A_2' machen, die k_1, k_2, σ als Parameter bekommen.

Wenn A_3 hat in A_1 und A_2 für k_1, k_2, σ die gleiche Erfolgswahrscheinlichkeit. Wenn A_3 Erfolg hat, dann auch A_1' oder A_2'.
Daraus folgt $\varepsilon \leq \varepsilon_1 + \varepsilon_2$. $\square$

HMAC

Als Anwendung des Thm ergeben sich HMACs.
Gegeben eine Hashfunktion $h : \{0,1\}^* \to \{0,1\}^b$.

HMAC von Nachricht m und Schlüssel k:

- HMAC $= h(k||opad||h(k||ipad||m))$. HMAC $= h(k \oplus opad||h(k \oplus ipad||m))$.
- $opad = 36 \cdots 36$.
- $ipad = 5C \cdots 5C$.

Die Benutzung von k anstelle von k_1 und k_2 basiert auf der Annahme, daß der „Unterschied“ von einem Angreifer aufgrund der Hashfunktionseigenschaften nicht bemerkt werden kann.

HMAC

Die Länge des Padding wird so eingestellt, daß eine volle Blocklänge der Kompressionsfunktion von h erreicht wird. Damit wird bei der zweiten Berechnung von h auch nicht intern iteriert.

Aufgrund von Geburtstagsangriffen ist die Sicherheit von HMAC bei iterierten Hashfunktionen trotzdem nur $2^{b/2}$.