Gottfried Wilhelm Leibniz Universität Hannover
Institut für Theoretische Informatik

Hash-basierte Nachrichten-Authentifizierungs-Codes

Bachelorarbeit

Chris Burmeister
Matrikelnr. 10019641

Hannover, den 18. August 2021

Erstprüfer: PD Dr. rer. nat. habil. Arne Meier
Zweitprüfer: Prof. Dr. rer. nat. Heribert Vollmer
Betreuer: PD Dr. rer. nat. habil. Arne Meier

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Bachelorarbeit/Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 18. August 2021

_____

Vorname Nachname

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In der modernen Welt sind Informationen wichtiger als je zuvor. Besonders im Internet werden Informationen beziehungsweise Daten gesammelt, weitergegeben und ausgewertet. Während die Daten den genutzten Internetseiten und Programmen zum größten Teil freiwillig zur Verfügung gestellt werden, gibt es auch andere, die sie sich einfach nehmen wollen. Um die unfreiwillige Weitergabe an Dritte zu verhindern, können die eigenen Nachrichten und Daten verschlüsselt werden.
Die Kryptographie ist die Wissenschaft, die sich genau damit, mit der Verschlüsselung und Sicherheit von Informationen, beschäftigt. Sie hat vier Grundziele: Vertraulichkeit, Integrität, Authentizität und Verbindlichkeit. Um diese Ziele umzusetzen, gibt es verschiedene Verfahren und Techniken.

Eine Technik ist die der Hashfunktionen. Sie dienen dazu, einen Text beliebiger Länge zu einem Text bestimmter Länge zu verschlüsseln. So versuchen diese vor allem das Ziel der Integrität umzusetzen, können aber auch der Vertraulichkeit dienen.
Eine weitere Technik ist die der digitalen Signaturen. Dabei wird eine Prüfsumme vom Sender an das Ende einer Nachricht angehängt und durch Überprüfen dieser, kann vom Empfänger festgestellt werden, ob die Nachricht unverändert ist und ob sie von einer bestimmten Person kommt.
Ein ähnliches Verfahren sind „Message Authentication Codes (MACs)“, welche den digitalen Signaturen ähnlich sind und auch eine Prüfsumme nutzen. Jedoch basieren diese auf einem symmetrischen Verschlüsselungsverfahren, während die digitalen Signaturen auf einem asymmetrischen Verfahren basieren. Trotz ihrer großen Ähnlichkeit sind MACs und digitale Signaturen eindeutig voneinander abzugrenzen.
MACs setzen mit Hilfe der Prüfsumme die Ziele der Integrität und Authentizität um. Sie haben mehrere Unterarten mit Relevanz, eine davon sind die "Hash-Based Message Authentication Codes (HMACs)". Diese vereinen die Idee der MACs mit dem Verfahren der Hashfunktionen.

In Kapitel 2 sind zuerst einige für diese Arbeit relevanten Notationseigenheiten zu finden. Danach gehe ich in Kapitel 3 auf die nötigen Grundlagen ein, welche für das Verständnis der hier behandelten Themen nötig sind.
Im darauffolgenden Kapitel 4 behandele ich die MACs im Allgemeinen. Ich gehe genau auf Funktion und Aufbau ein und stelle einige Arten von MACs vor. In Kapitel 5 folgt der um-

fangreichste Teil, welcher sich mit auf Hashfunktionen basierenden MACs beschäftigt. Dort werden neben den HMACs, welche Hauptthema dieser Arbeit sind, auch kurz die NMACs betrachtet. Außerdem gehe ich auf Aufbau und Benutzung der HMACs, sowie auf ihre Sicherheit, Angriffe gegen sie und relevante Modifikationen ihrer Struktur ein. Im abschließenden Kapitel 6 fasse ich die Ergebnisse dieser Arbeit im Fazit noch einmal zusammen und gebe einen Ausblick auf aktuelle Entwicklungen und weitere interessante Themen.

2 Notationen

A^ℓ bezeichnet die Menge aller Zeichenfolgen der Länge ℓ, bestehend aus Zeichen aus der Menge A.
$\{0,1\}^\ell$ bezeichnet die Menge aller Zeichenfolgen der Länge ℓ, bestehend aus 0 und 1.
$\{0,1\}^{\ell+}$ bezeichnet die Menge aller Zeichenfolgen der Länge ℓ oder länger, bestehend aus 0 und 1.
$\{0,1\}^{<\ell}$ bezeichnet die Menge aller Zeichenfolgen mit kürzerer Länge als ℓ, bestehend aus 0 und 1.
$\{0,1\}^{\ell*}$ bezeichnet die Menge aller Zeichenfolgen mit einer Länge, die ein vielfaches von ℓ ist, bestehend aus 0 und 1.
$\{0,1\}^*$ bezeichnet die Menge aller Zeichenfolgen mit beliebiger Länge, bestehend aus 0 und 1.

$Pr[X = x]$ bezeichnet die Wahrscheinlichkeit dafür, dass die Zufallsgröße X den Wert x annimmt.

$x \oplus y$, wobei x und y Zeichenfolgen aus mehreren Bits sind, bezeichnet die XOR-Verknüpfung der einzelnen Bits, der beiden Zeichenfolgen.

$x \cdot y$ bezeichnet die Zeichenfolge bestehend aus der Zeichenfolge x mit der hinten angehängtem Zeichenfolge y.

$a|b$ bezeichnet die Eigenschaft, dass die Zahl a die Zahl b teilt, bzw. dass b ein vielfaches von a ist.

$(x)_2^r$ bezeichnet die Binärdarstellung von $|x|$, welche durch vorangestellte Nullen auf Länge r erweitert wurde.

3 Grundlagen

3.1 Allgemeine Grundlagen

Ziele der Kryptographie

Die Kryptographie hat vier Ziele, deren Erfüllung für die sichere Übertragung von Daten sorgt. Die vielen verschiedenen kryptographischen Verfahren setzen meist mindestens eines dieser Ziele um, eventuell aber auch mehrere.

Die Ziele sind folgende:

Vertraulichkeit

Eine Nachricht soll nur von dem gewünschten Empfänger gelesen werden können. Dritten soll es nicht möglich sein, Auskunft über den Inhalt der Nachricht zu erlangen.

Integrität

Die Nachricht soll nicht von Dritten in irgendeiner Weise verändert werden, ohne dass diese Änderung bemerkt wird.

Authentizität

Es muss möglich sein, für den Absender einer Nachricht gegenüber dem Empfänger zu beweisen, dass er die Nachricht geschickt hat. Diese Urheberschaft sollte nicht angezweifelt und nicht von Dritten gefälscht werden können.

Verbindlichkeit

Die Verbindlichkeit ist der Authentizität sehr ähnlich. Auch bei ihr geht es um die Urheberschaft, die bewiesen werden soll. Der Unterschied ist, dass es dem Urheber hierbei nicht möglich sein soll, die Urheberschaft, auch gegenüber Dritten, abzustreiten.

Kryptosystem

Ein System, das der Ver- und Entschlüsselung von Texten mit Hilfe eines Schlüssels (Key) dient, nennt man Kryptosystem. Es ist der Grundstein für sichere Kommunikation und der meisten Verfahren in der Kryptographie. Die mathematische Definition nach Küsters und Wilkes Lehrbuch ([KW11]) lautet wie folgt:

Definition 3.1 (Kryptosystem)**.** Ein *Kryptosystem* (crypto system) ist ein Tupel $\varphi = (X, K, Y, e, d)$. Wobei X die nicht leere Menge der Klartexte, K die endliche Menge der Schlüssel und Y die Menge der verschlüsselten Texte / Chiffrentexte ist.
e ist eine Verschlüsselungsfunktion, $e: X \times K \to Y$ und d ist die zugehörige Entschlüsselungsfunktion, $d: Y \times K \to X$, sodass $d(e(x,k),k) = x$ für alle $x \in X$, $k \in K$ und $Y = \{e(x,k) | x \in X, k \in K\}$ erfüllt ist.

Symmetrische Verschlüsselungsverfahren

Bei einem symmetrischen Verschlüsselungsverfahren versuchen beispielsweise zwei Personen miteinander über einen Kanal zu kommunizieren. Die übertragenen Nachrichten werden dafür verschlüsselt. Beide Personen nutzen bei einem symmetrischen Verfahren den gleichen Schlüssel zum ver- und entschlüsseln. Damit Dritte die Nachrichten nicht auch lesen können, muss dieser Schlüssel also zwingend geheim bleiben. Zudem muss vor der eigentlichen Kommunikation ein Weg gefunden werden, den Schlüssel unter den zwei Personen auszutauschen. Die Personen müssen sich auch zuvor auf einen Schlüssel festlegen, wodurch zwei Personen nicht ohne vorherige Planung geheim miteinander kommunizieren können. Die Definition nach Küsters und Wilkes Lehrbuch ([KW11]) ist:

Definition 3.2 (symmetrisches Kryptoschema)**.** Es sei $\ell > 0$. Ein *symmestrisches ℓ-Kryptoschema* ist ein Tupel $S = (K, E, D)$, bestehend aus einer Schlüsselmenge $K \subseteq \{0,1\}^s$ für ein $s \geq 1$, einem zufallsgesteuerten Chiffrieralgorithmus $E(x: \{0,1\}^{\ell *}, k: K): \{0,1\}^*$ und einem deterministischen Dechiffrieralgorithmus $D(y: \{0,1\}^*, k: K): \{0,1\}^{\ell *}$.
Dabei müssen die Laufzeiten von E und D polynomzeitbeschränkt sein in der Länge von x bzw. y, d. h., es existieren Polynome p und q, so dass die Laufzeiten von $E(x,k)$ und $D(y,k)$ für alle x, y und k beschränkt sind durch $p(|x|)$ bzw. $q(|y|)$. Außerdem muss für jeden Klartext $x \in \{0,1\}^{\ell *}$, jede Zufallsfolge $\alpha \in \{0,1\}^{p(|x|)}$ und jeden Schlüssel $k \in K$ gelten:
$D(E^\alpha(x,k),k) = x$

Asymmetrisches Verschlüsselungsverfahren

Bei einem asymmetrischen Verschlüsselungsverfahren besitzt jede Person zwei Schlüssel, einen öffentlichen und einen privaten. Möchte nun eine Person einer anderen eine geheime Nachricht schicken, nutzt sie zum Verschlüsseln der Nachricht den öffentlichen Schlüssel des Empfängers. Dieser kann mit Hilfe seines privaten Schlüssels nun die empfangene Nachricht entschlüsseln. Eine mit einem der öffentlichen Schlüssel verschlüsselte Nachricht kann nur mit dem zugehörigen privaten Schlüssel entschlüsselt werden. Da der private Schlüssel nie übertragen werden muss, ist es für einen Dritten praktisch unmöglich, die Nachricht zu lesen, da dies so viel Rechenaufwand bedeuten würde, dass es nicht in Polynomialzeit machbar ist.

Die Definition nach Küsters und Wilkes Lehrbuch ([KW11]) ist:

Definition 3.3 (asymmetrisches Kryptosschema)**.** Ein *asymmetrisches Kryptoschema* ist ein Tupel $S = (X, K, G, E, D)$ bestehend aus einem Klartextraum X, einer Schlüsselmenge K, einem zufallsgesteuerten Schlüsselgenerierungsalgorithmus $G : K_{pub} \times K_{priv}$, einem zufallsgesteuerten Chiffrieralgorithmus $E(x\colon X, k\colon K_{pub})\colon \{0,1\}^*$ und einem deterministischen Dechiffrieralgorithmus $D(y\colon \{0,1\}^*, \hat{k} : K_{priv}) : \{0,1\}^*$,
wobei K, K_{pub} und K_{priv} wie folgt definiert sind: K ist die Menge der von G gelieferten Ausgaben. Jedes Element von K ist ein Paar von Bitvektoren, das Schlüsselpaar genannt und meistens mit $(k, \hat{k})$ bezeichnet wird. Dabei ist k ein öffentlicher und $\hat{k}$ ein privater Schlüssel. Die Menge der öffentlichen Schlüssel wird mit K_{pub}, die Menge der privaten Schlüssel mit K_{priv} bezeichnet. Es wird verlangt, dass die Laufzeit von G durch eine Konstante beschränkt ist. Die Laufzeiten von E und D sollen in der Länge der Eingaben polynomiell beschränkt sein, d. h., es existieren Polynome p und q, so dass, für alle $x \in X, y \in \{0,1\}^*$ und $(k, \hat{k}) \in K$, die Laufzeiten von $E(x, k)$ und $D(y, \hat{k})$ durch $p(|x|)$ bzw. $q(|y|)$ beschränkt sind. Außerdem muss gelten:
$D(E^{\alpha}(x, k).\hat{k})$=x für alle $x \in X, (k, \hat{k}) \in K$ und $\alpha \in \{0,1\}^{p(|x|)}$)

Blockverschlüsselung / Blockkryptosystem

Blockverschlüsselung ist ein symmetrisches kryptographisches Verschlüsselungsverfahren. Dabei werden Textblöcke einer festen Länge mit Hilfe eines Schlüssels zu Blöcken mit fester Länge verschlüsselt. Um längere Nachrichten (welche die Blocklänge überschreiten) zu verschlüsseln, wird der Text in mehrere Teile/Blöcke eingeteilt und nacheinander verschlüsselt. Es ist eine Funktion erforderlich, welche den Text auf diese Weise verschlüsselt. Diese Funktion kann in verschiedenen Betriebsmodi arbeiten. Ein wichtiger Modus ist der Cipher-Block-Chaining Modus, bei welchem die einzelnen Textblöcke mit dem jeweils vorherigen XOR-Verknüpft werden bevor sie verschlüsselt werden. Die mathematische Definition eines allgemeinen Blockkryptosystems nach Küsters und Wilkes Lehrbuch ([KW11]) sieht wie folgt aus:

Definition 3.4 (Block-Kryptosystem)**.** Es sei $\ell > 0$. Ein *ℓ-Block-Kryptosystem* B ist ein Kryptosystem der Form: $B = (\{0,1\}^{\ell}, K, \{0,1\}^{\ell}, E, D)$ mit $K \subseteq \{0,1\}^{s}$ für ein $s > 0$.
E und D sind Algorithmen zu Ver-(E) und Entschlüsselung(D).

Beispiel 3.1 (Blockkryptosystem)**.** Es wird ein ℓ-Block-Kryptosystem $B_{10} = (\{0,1\}^{\ell}, K, \{0,1\}^{\ell}, E, D)$ definiert. Es sei $\ell, s = 10$. Die Ver- und Entschlüsselungsalgorithmen seien durch
$e(x, k) = x \oplus k$ und $d(y, k) = y \oplus k$ für alle $x, y, k \in \{0,1\}^{\ell}$.
Beispielsweise ergibt sich für die Eingabe x = 1011101000 mit k = 0111010110 e(x,k) = 1100111110.

Padding

Padding bezeichnet Fülldaten, welche dazu dienen, Daten in bestimmte Form zu bringen. So wird Padding häufig benutzt, um eine Bitfolge zu verlängern. Dazu werden beispielsweise an den Anfang oder das Ende einer Bitfolge Nullen angefügt und mit einer Eins von den anderen Bits getrennt. Diese Art des Paddings wird in dieser Arbeit *simples Padding* genannt. Die Bits, die durch das simple Padding hinzugefügt werden, beinhalten dabei keine weiteren Informationen und dienen nur dem Anpassen der Bitfolge auf eine bestimmte Länge. Es gibt auch Paddingarten, welche die hinzugefügten Bits nutzen, um weitere Informationen zu codieren. Eine Funktion, die Padding erzeugt, nennt man Füllfunktion.

Beispiel 3.2 (simples Padding)**.** Es soll einen Zeichenfolge $x = 1010$ auf eine Länge von zehn gebracht werden. Die mit simplen Padding verlängerte Zeichenfolge x' sieht wie folgt aus $x' = 1010100000$.

Famile von Funktionen

Eine *Familie von Funktionen* (oder auch Funktionsfamilie) ist eine Menge mehrerer Funktionen, welche sich nur im Initialisierungsvektor / Schlüssel unterscheiden. Solche Variablen werden in diesem Kontext auch Index genannt. Der Definitionsbereich und der Aufbau der Funktionen sind in einer Familie gleich.
F_k mit $k \in K$ bezeichne die Familie F mit Schlüssel (oder auch Index) k aus der Indexmenge K. Eine Familie von Funktionen kann auch als eine einzige Funktion angesehen werden, welche den Index k als weiteren Input bekommt.

Pseudozufällige Funktionsfamilie

Eine Funktionsfamilie F_n kann als *pseudozufällig* oder auch als PRF bezeichnet werden, wenn für jedes n bei Betrachtung von Eingabe x und Ausgabe y die Funktion nicht von einer wirklich zufälligen Funktion unterscheidbar ist.
Die Definition auf Grundlage von Goldreichs Buch *Foundations of Cryptographie* ([Gol03]):

Definition 3.5 (Pseudozufällige Funktionsfamilie)**.** Ein ℓ-bit Funktionsfamilie $F = \{F_n\}_{n\in\mathbb{N}}$ heißt *pseudozufällig*, wenn für alle probabilistischen polynomialzeit Orakelmaschinen M, für alle polinome $p(\cdot)$ und für alle genügend langen n gilt:
$|Pr[M^{F_n}(1^n) = 1] - Pr[M^{H_n}(1^n) = 1]| < \frac{1}{p(n)}$,
wobei $H = \{H_n\}_{n\in\mathbb{N}}$ die einheitliche ℓ-bit Funktionsfamile ist.

Diese Definition bedarf ohne den Kontext von Goldreichs Buch noch einiger Erklärungen. Eine ℓ-bit Funktionsfamilie bezeichnet eine Funktionsfamilie welche von ℓ-bit langen Zeichenketten auf ℓ-bit lange Zeichenketten abbildet. Die einheitliche ℓ-bit Funktionsfamilie $H = \{H_n\}_{n\in\mathbb{N}}$ ist demnach eine ℓ-bit Funktionsfamilie, deren Funktionen gleichmäßig über den Raum aller ℓ-bit Funktionsfamilien verteilt sind.

Eine Orakelmaschine ist eine Turingmaschine, die für eine bestimmten Input x mit Hilfe einer Funktionsfamilie F eine Entscheidung y darüber trifft, ob x zu dieser Funktionsfamilie gehört. Es gilt $M^F(x) = y$. Für eine genauere Definition siehe Goldreichs *Foundations of Cryptographie* ([Gol03]) Definiton 1.3.8.
Die Eigenschaft eine Pseudozufällige Funktionsfamilie zu sein, ist häufig für die Sicherheit kryptographischer Verfahren im Bereich der symmetrischen Authentifizierungsschemata notwendig, welche später in dieser Arbeit thematisiert werden.

3.2 Hashfunktionen

Eine Hashfunktion im kryptographischen Zusammenhang ist eine Funktion die eine Zeichenfolge beliebiger Länge in eine verschlüsselte Zeichenfolge bestimmter Länge wandelt. Sie werden meistens zum Erreichen des Ziels der Integrität genutzt. Es gibt verschiedene Konstruktionen für Hashfunktionen. Eine, welche in dieser Arbeit relevant ist, ist die Merkle-Darmgård-Konstruktion ([Mer79] & [Dam90]). Die mathematische Definition einer allgemeinen Hashfunktion, nach Küsters und Wilkes Lehrbuch ([KW11]), sieht wie folgt aus:

Definition 3.6 (Hashfunktion)**.** Es seien $\ell > 0$ und $\{0,1\}^\ell \subset D \subseteq \{0,1\}^*$. Eine Funktion $h : D \to \{0,1\}^\ell$ heißt *Hashfunktion* im Definitionsbereich D. Ist $D = \{0,1\}^{\leq L}$ für $L > \ell > 0$, so wird h eine *(L, ℓ)-beschränkte Hashfunktion* oder einfach beschränkte Hashfunktion genannt. Für $D = \{0,1\}^*$ heißt h unbeschränkte ℓ-Hashfunktion oder einfach unbeschränkte Hashfunktion. Die Zahl ℓ ist die Hashbreite. Der Bitvektor $h(x)$ wird Hashwert von x genannt.

Kollisionsresistenz

Eine *Kollision* ist der Fall in dem zwei unterschiedliche Eingabewerte die gleichen Hash-/Ausgabewerte erzeugen. Die Eingabewerte x, y erzeugen eine Kollision für die Hashfunktion h, wenn $x \neq y$ und $h(x) = h(y)$ gilt. Wird solch eine Kollision in einem Verfahren, das auf Hashfunktionen basiert, gefunden, beeinträchtigt dies die Sicherheit des Verfahrens. Deswegen wird von solchen Verfahren Kollisionsresistenz verlangt.
Die *Kollisionsresistenz* ist die Fähigkeit dem Finden von Kollisionen zu widerstehen. Da der Definitionsbereich von Hashfunktionen nach der obigen Definition größer als ihr Wertebereich ist, ist es unmöglich, eine Hashfunktion zu finden, welche keinerlei Kollisionen besitzt. Daher wird gefordert, dass es für einen ressourcenbeschränkten Algorithmus nur mit sehr geringer Wahrscheinlichkeit möglich sein soll, eine Kollision in Polynomialzeit auszugeben.
Es wird weiterhin unterschieden zwischen schwacher und starker Kollisionsresistenz. Während bei *schwacher Kollisionsresistenz* Schutz gegen eine Kollision zu einer vorgegebenen Eingabe verlangt wird, wird bei *starker Kollisionsresistenz* Schutz gegen eine beliebige Kollision verlangt. Es ist für einen Algorithmus deutlich schwerer, eine Kollision bei einer bestimmten Eingabe zu finden. Da bei diesem Sicherheitsbegriff mehr Anforderungen an

den Angreifer/Algorithmus gestellt werden, ist der daraus resultierende Sicherheitsbegriff schwächer und wird somit schwache Kollisionsresistenz genannt.
Formal sind die beide Sicherheitsbegriffe in einem Paper von P. Rogaway und T. Shrimpton definiert (siehe [RS04]).

Familie von Hashfunktionen

Eine *Familie von Hashfunktionen* H_k wird wie eine Familie von Funktionen definiert. Die Definition wird lediglich um die Bedingung erweitert, dass es sich bei den Funktionen um Hashfunktionen handeln muss.

Geburtstagsparadoxon

Das *Geburtstagsparadoxon* bezeichnet die Tatsache, dass bei der Betrachtung von 23 Personen bereits eine über 50-prozentige Wahrscheinlichkeit besteht, zwei Geburtstage am gleichen Tag zu haben. Die Chance darauf wird meistens deutlich kleiner geschätzt und erscheint deswegen paradox. Die Chance ist aus dem Grund so hoch, da für jede Person, die hinzukommt, deutlich mehr mögliche Paare entstehen, denn aus der Personengruppe muss nur irgendein Paar den gleichen Geburtstag haben. Die Formel zur Berechnung einer solchen Wahrscheinlichkeit ergibt sich aus der Formel von Laplace:

$$p = 1 - \frac{n!}{(n-k)! \cdot n^{\mathrm{k}}}$$

In diesem konkreten Fall wäre $n = 365$ die Anzahl der möglichen Geburtstage und $k = 23$ die Anzahl der Personen, die betrachtet werden.
Das Wissen über dieses Paradoxon ermöglicht nun eine Methode, um bei einer Hashfunktion Kollisionen zu finden. Dies wird als *Geburtstagsangriff* bezeichnet. Ähnlich wie beim Paradoxon werden bei diesem Angriff eine bestimmte Menge an zufälligen Klartexten genommen und zu ihren Hashwerten umgewandelt. Danach werden alle so erzeugten Hashs miteinander verglichen, um festzustellen, ob eine Kollision gefunden wurde. Da bei modernen Hashfunktionen die Hashbreite bei mindestens 128 Bits liegt, ergibt sich ein k an zu berechnenden und zu vergleichenden Werten von ungefähr 2^{64}. Dadurch ist der Geburtstagsangriff nicht sehr effektiv, weil sehr viel Zeit benötigt wird, um so viele Berechnungen durchzuführen. Die benötigte Zeit ist ohne einen Quantencomputer zu hoch, als dass dies ein effektiver Angriff wäre.
Da dieser Angriff aber auf allen Hashfunktionen mit genügend Zeit möglich ist, bildet er eine obere Schranke für die Kollisionsresistenz einer Hashfunktion. Demnach sollte eine Hashfunktion so aufgebaut sein, dass es keinen effektiveren Angriff als den Geburtstagsangriff gegen sie gibt und sollte sich mit einer genügend langen Hashbreite auch gegen diesen schützen.

Merkle-Damgård-Konstruktion

Die *Merkle-Damgård-Konstruktion* (kurz MD) ist, wie bereits erwähnt, eine Möglichkeit eine Hashfunktion aufzubauen. Diese wird heutzutage in mehreren Verfahren verwendet (siehe [KW11]). Beispiele dafür sind die bekannten Hashfunktionen md5, SHA-1 und Sha-2. Die Konstruktion geht auf Ralph C. Merkle (siehe [Mer79]) und Ivan Damgård (siehe [Dam90]) zurück. In dieser Konstruktion wird die Hashfunktion durch das wiederholte Anwenden einer sogenannten *Kompressionsfunktion* realisiert. Die formale Definition einer Kompressionsfunktion nach Küsters und Wilkes Lehrbuch ([KW11]) ist wie folgt:

Definition 3.7 (Kompressionsfunktion)**.** Es seien $\ell, b > 0$. Eine *(ℓ, b)-Kompressionsfunktion* ist eine Funktion $f\colon \{0,1\}^{\ell} \times \{0,1\}^{\mathrm{b}} \to \{0,1\}^{\ell}$. Eine solche Funktion wird auch aufgefasst als Funktion der Form $f\colon \{0,1\}^{\ell+b} \to \{0,1\}^{\ell}$. Dabei nennt man ℓ die Kompressionslänge und b die Blocklänge von f. Eine Kollision von f ist ein Paar $(x, x') \in \{0,1\}^{\ell+b} \times \{0,1\}^{\ell+b}$, für das $x \neq x'$ und f(x) = f(x') gilt.

Beispiel 3.3 (Kompressionsfunktion)**.** Es wird eine Kompressionsfunktion $f_{xor}(x, h) = x_1 \ldots x_{10} \oplus h_1 \ldots h_{10}$ mit ℓ und $b = 10$ definiert. Diese Kompressionsfunktion ist nicht kollisionsresistent und kann somit in einem sicheren Verfahren nicht genutzt werden.

Dieser iterative Ablauf benötigt ähnlich wie bei einem Blockkryptosystem Blöcke einer bestimmten Größe. In diese Blöcke wird also der zu verschlüsselnde Klartext zu Beginn aufgeteilt. Auf die Blöcke wird die Funktion nun nacheinander angewendet, wobei ihr jeweils der aktuelle Block und das Ergebnis des vorherigen Blocks als Eingabe dienen (siehe Abbildung 2.1 im Folgenden). Um diese Konstruktion auch verwenden zu können, wenn der Eingabetext kein Vielfaches von b ist, wird Padding benötigt. Jedoch wird hier nicht einfach das zuvor beschriebene simple Padding benutzt. Es wird eine Füllfunktion, welche bestimmte Eigenschaften erfüllen muss, sodass sie kompatibel für die Merkle-Damgård-Konstruktion (MD) ist, benutzt. Mit anderen Worten: Damit es eine *MD-kompatible Füllfunktion* ist. Diese Kompatibilität ist notwendig, damit sich die Kollisionsresistenz von der Kompressionsfunktion auf die gesamte Hashfunktion überträgt.
Formal ist eine MD-kompatible Füllfunktion durch Küsters und Wilkes Lehrbuch ([KW11]) wie folgt definiert:

Definition 3.8 (MD-kompatible Füllfunktion)**.** Es seien r und b natürliche Zahlen mit $0 < r \leq b$. Eine Funktion $p\colon \{0,1\}^{<2^r} \to \{0,1\}^{b+}$ heißt *MD-kompatible* (r, b)*-Füllfunktion*, falls die folgenden Bedingungen erfüllt sind:
1. Für jedes $x \in \{0,1\}^{2^r}$ ist x ein Präfix von $p(x)$.
2. Für alle $x_0, x_1 \in \{0,1\}^{2^r}$ mit $|x_0| = |x_1|$ gilt $|p(x_0)| = |p(x_1)|$.
3. Für alle $x_0, x_1 \in \{0,1\}^{2^r}$ mit $|x_0| \neq |x_1|$ ist der Suffix von $p(x_0)$ der Länge b verschieden vom Suffix von $p(x_1)$ der Länge b.

Eine Füllfunktion ist also MD-kompatibel, wenn die eigentliche Nachricht (x) nicht verändert (nur erweitert) wird, gleich lange Eingaben in die Funktion gleich lange Ausgaben ergeben und bei unterschiedlich langen Eingaben unterschiedlich erweitert wird.
Ist eine Füllfunktion MD-kompatibel, so sorgt dies wie bereits erwähnt dafür, dass die Kollisionsresistenz der Kompressionsfunktion sich auf die gesamte Hashfunktion überträgt. (nach Küsters und Willkes Lehrbuch [KW11])

Beispiel 3.4 (MD Füllfunktion)**.** Im Folgenden bezeichnet $(x)_2^r$ die Binärdarstellung von $|x|$ auf die Länge r erweitert durch das Voranstellen von Nullen. Es seien $r = 4$ und $b = 10$ und die Füllfunktion $p_{MD}^{r,b}(x) = p_{MD}^{4,10}(x)$ sei gegeben durch $p_{MD}^{4,10}(x) = x \cdot 1 \cdot 0^s \cdot (x)_2^r$, wobei $s \geq 0$ minimal so gewählt wird, dass $b - r = (|x| + 1 + s) \bmod b$, bzw. in diesem konkreten Fall $6 = (|x| + 1 + s) \bmod 10$ gilt. Diese Füllfunktion weist alle Eigenschaften auf, um als MD-kompatibel zu gelten:
x wird nicht verändert und nur nach hinten erweitert.
Zwei verschiedene Eingaben gleicher Länge ergeben gleich lange Erweiterungen.
Zwei unterschiedlich lange Eingaben ergeben verschiedene Erweiterungen.
Bei der Eingabe $x = 1010$ wird der Text durch diese Funktion wie folgt erweitert: $p(x) = 1010100100$.
Bei der Eingabe $x = 1010001$ wird der Text zu $p(x) = 10100011000000000111$.

Die Hashfunktion, die durch das wiederholte Anwenden einer Kompressionsfunktion entsteht, wird auch *Iterationsfunktion* genannt. Es wird nach Küsters und Wilkes Lehrbuch ([KW11]) wie folgt definiert:

Definition 3.9 (Iterationsfunktion)**.** Es sei f eine (ℓ, b)-Kompressionsfunktion. Dann ist die zugehörige *Iterationsfunktion* $i^{\mathrm{f}} : \{0,1\}^\ell \times \{0,1\}^{\mathrm{b}*} \to \{0,1\}^\ell$ induktiv definiert durch:

$$
\begin{aligned}
&i^{\mathrm{f}}(u, \varepsilon) = u && \text{für alle } u \in \{0,1\}^\ell,\\
&\mathrm{i}^{\mathrm{f}}(u, xv) = f(i^{\mathrm{f}}(u, x), v) && \text{für alle } u \in \{0,1\}^\ell, x \in \{0,1\}^{\mathrm{b}*}, v \in \{0,1\}^{\mathrm{b}}
\end{aligned}
$$

Mit den zuvor beschrieben Funktionen kann nun eine Hashfunktion nach dem Merkle-Damgård-Prinzip durch Küsters und Wilkes Lehrbuch ([KW11]) wie folgt definiert werden:

Definition 3.10 (Hashfunktion nach dem Merkle-Damgård-Prinzip)**.** Es sei f eine (ℓ, b)-Kompressionsfunktion, $p : \{0,1\}^{<2^{\mathrm{r}}} \to \{0,1\}^{\mathrm{b}+}$ eine MD-kompatible (r, b)-Füllfunktion und $u \in \{0,1\}^\ell$ ein Initialisierungsvektor. Die nach dem Merkle-Damgård-Prinzip definierte iterierte *MD-Hashfunktion* $h_u^{f,p} : \{0,1\}^{<2^{\mathrm{r}}} \to \{0,1\}^\ell$ ist definiert durch
$h_u^{f,p} = i^{\mathrm{f}}(u, p(x))$.

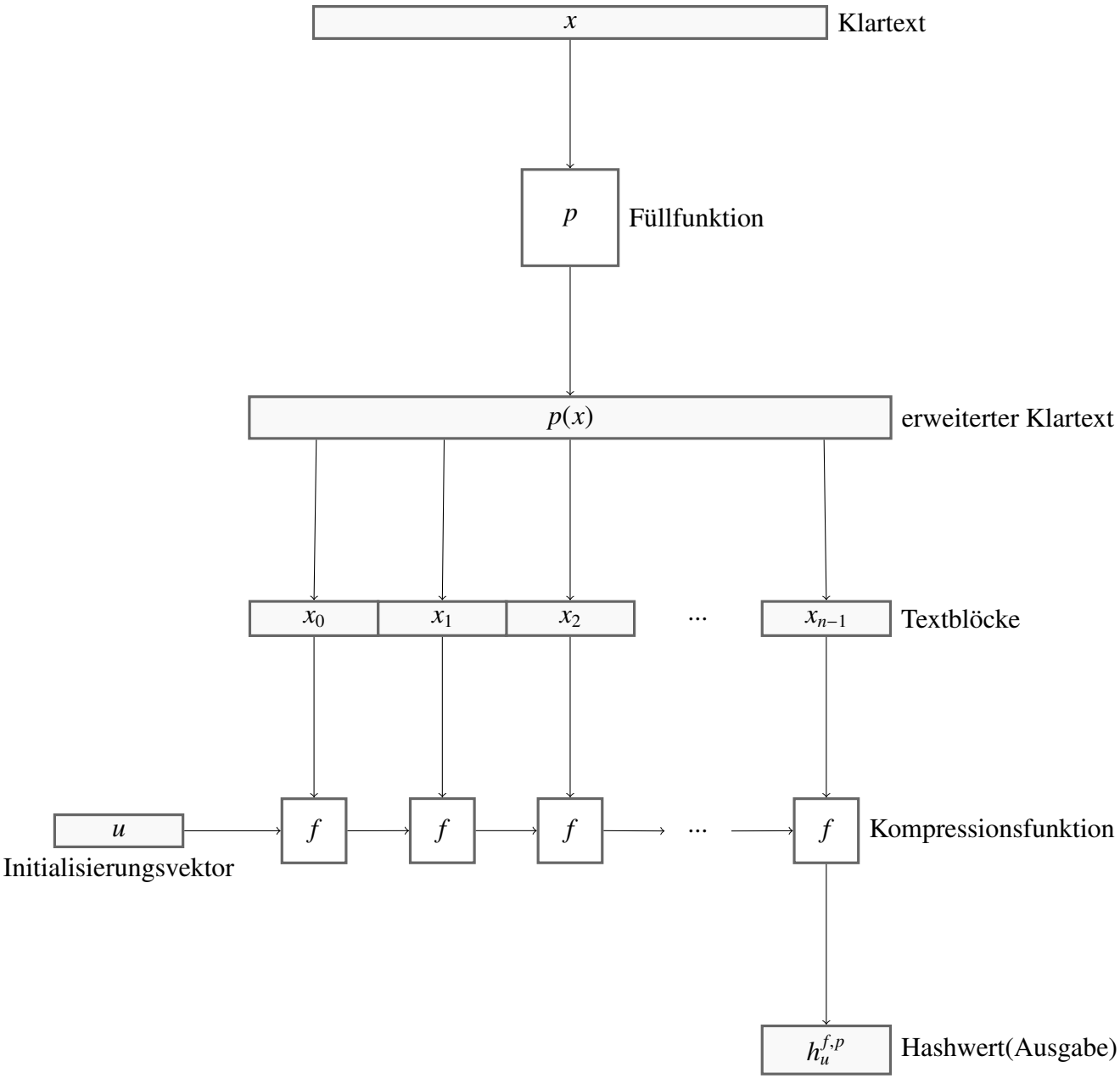

Abbildung 3.1: Merkle-Damgård-Hashfunktion

Beispiel 3.5 (simple MD-Hashfunktion)**.** Sei f die in Beispiel 3.3 beschriebene simple Kompressionsfunktion $f = f_{xor}$ und p die in Beispiel 3.4 beschriebene Fullfunktion $p = p_{MD}^{4,10}$. Es sei weiterhin $u = 1111111111$.
Die Hashfunktion nach dem Merkle-Damgård Prinzip ist dann gegeben durch $h_{1111111111}^{f_{xor},p_{MD}^{4,10}}(x) = h_u^{f,p}(x) = i^f(u, p(x))$. Wobei nach Definition 3.9 natürlich $i^f(u, \varepsilon) = u$ und $i^f(u, xv) = f(i^f(u, x), v)$ gilt.
Für die beispielhafte Eingabe $x = 101010101010$ wird nun wie folgt verfahren:
Zuerst wird von der Füllfunktion p x auf $p(x) = 10101010101010001100$ erweitert.
Dann wird diese Zeichenfolge in die zwei Blöcke der Länge b aufgeteilt $x_0 = 1010101010$ und $x_1 = 1010001100$.
Diese werden dann iterativ mit der Kompressionsfunktion verschlüsselt, sodass $f(u, x_0) = 0101010101 = y_0$ und $f(y_0, x_1) = 1111011001$, also $h_u^{f,p}(x) = i^f(u, p(x)) = f(f(u, x_0), x_1) = 1111011001$.

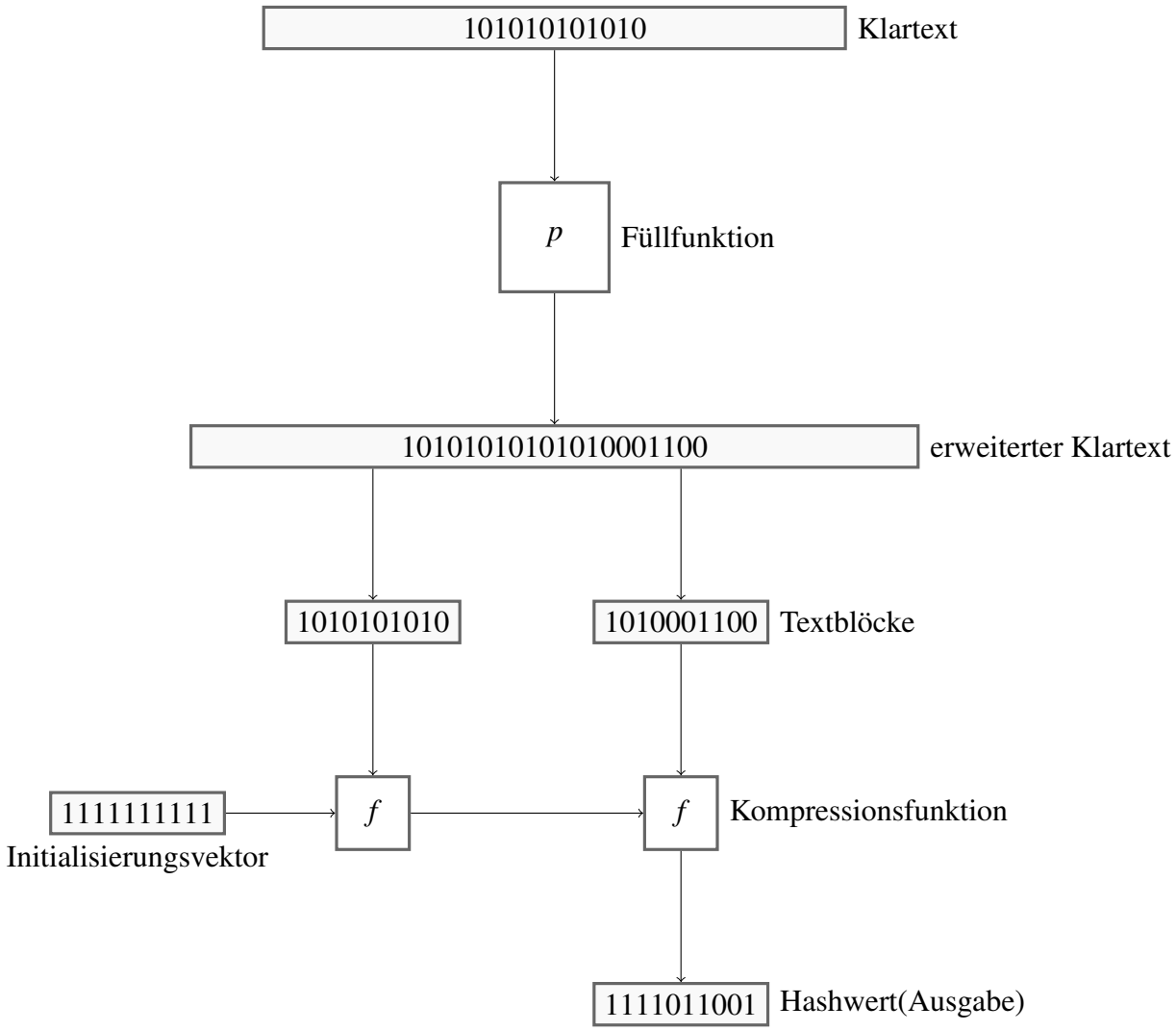

Abbildung 3.2: Merkle-Damgård-Hashfunktion mit Werten aus Beispiel 3.5

4 Symmetrische Authentifizierungsverfahren

Symmetrische Authentifizierungsverfahren werden auch MACs genannt. Dabei steht MAC für Message Authentication Code oder auf deutsch Nachrichtenauthentifizierungscode. Ein solcher Code wird bei der Übertragung von Daten genutzt. Der Sinn eines Authentifizierungsverfahren ist es, Integrität und Authentizität einer Nachricht überprüfen zu können, wozu bei einem solchen Verfahren eine Prüfsumme beziehungsweise der bereits erwähnte Code oder wie im Folgenden auch ein Etikett erstellt wird. Dieses wird dann an die Nachricht, die übermittelt werden soll, angehängt und mit der Nachricht versendet. Der Empfänger kann nun im Zuge des Verfahrens das Etikett auswerten und so Integrität und Authentizität überprüfen. Bei Authentifizierungsverfahren wird zwischen zwei Typen unterschieden, den symmetrischen Authentifizierungsverfahren, welche auf symmetrischen Verschlüsselungsverfahren basieren und hier Thema sind und den asymmetrischen Authentifizierungsverfahren, welche wiederum auf asymmetrischen Verschlüsselungsverfahren basieren. Erstere sind wie bereits erwähnt eher als Message Authentication Codes (MAC) bekannt, während die asymmetrischen Authentifizierungsverfahren unter Digitale Signaturen geläufig sind. Beide haben Vor- und Nachteile dem anderen gegenüber, jedoch ist leicht aus dem Thema dieser Arbeit zu erkennen, dass im Folgenden auf das symmetrische Verfahren, die MACs, eingegangen wird. Die Digitalen Signaturen werden im Folgenden nicht weiter behandelt.

4.1 Aufbau eines MACs

Es gibt viele verschiedene Arten MACs zu erzeugen, mit verschiedenen Komplexitäten. Meistens hat ein komplizierterer Aufbau eine größere Sicherheit zum Ziel. Einige der verschiedenen Konstruktionen haben sogar eigene Namen und bilden Unterarten der MACs. Eine dieser Unterarten ist der HMAC, der Hash-Based Message Authentication Code, welcher das Hauptthema dieser Arbeit ist und auf den im nächsten Kapitel eingegangen wird.
In diesen Kapitel werden jedoch zunächst die MACs im Allgemeinen und die Konstruktionen, die nicht auf Hashfunktionen basieren, behandelt.

4.1.1 Allgemeiner Aufbau

Im Folgenden wird ein allgemeines MAC-Schema nach Küsters und Willkes Lehrbuch ([KW11]) definiert:

Definition 4.1 (MAC)**.** Es sei $\ell > 0$ und $D \subseteq \{0,1\}^*$ Ein (D, ℓ)-MAC ist ein Tupel $M = (K, T)$. Dabei ist D die Menge der möglichen Nachrichten, $K \subseteq \{0,1\}^*$ die endliche Menge der Schlüssel und T ein deterministischer Etikettieralgorithmus $T(x\colon D, k\colon K)\colon \{0,1\}^\ell$. Ein Etikett t ist gültig für k, wenn $T(x,k) = t$ gilt. (x,t) wird dann als gültiges Nachrichten-Etikett-Paar bezüglich k bezeichnet.

Ein Etikettieralgorithmus ist ein Algorithmus, der aus Schlüssel und Nachrichtentext das Etikett erstellt. Der genaue Aufbau dieses Algorithmus ist je nach Konstruktion des MACs unterschiedlich.
Wie bereits angedeutet, würde in einer Anwendung (x,t) beziehungsweise $(x, T(x,k))$ übermittelt werden. Um nun das Etikett zu validieren, ist noch ein Validierungsalgorithmus V nötig. Dieser ist in diesem Fall trivial:
Um zu überprüfen, ob (x,t) ein gültiges Nachrichten-Etikett-Paar zu k ist, kann einfach ein neues Etikett aus x und k erstellt und dann mit t verglichen werden. Bei Gleichheit ist (x,t) gültig.

4.1.2 Aufbau aus Blockkryptosystemen

Eine Möglichkeit, eine MAC-Konstruktion zu erzeugen, ist es, Blockkryptosysteme als Grundlage zu nutzen. Die in den Grundlagen aufgestellte Definition 3.4 eines Blockkryptosystems kann leicht so angepasst werden, dass daraus ein MAC entsteht (basierend auf Küsters und Wilkes Lehrbuch ([KW11])):

Definition 4.2 (MAC aus Blockkryptosystem)**.** Sei $B = (\{0,1\}^\ell, K, \{0,1\}^\ell, E, D)$ mit $K \subseteq \{0,1\}^s$ für $\ell, s > 0$ ein ℓ-Blockkryptosystem. Der MAC aus B, sei $M_\mathrm{B} = (K, T)$, mit $T(x,k) = E(x,k)$.

Der Etikettieralgorithmus T ist also in dieser Konstruktion durch den Verschlüsselungsalgorithmus E des Blockkryptosystems gegeben. Der Validierungsalgorithmus kann dann natürlich leicht durch den Entschlüsselungsalgorithmus D realisiert werden oder wie im Allgemeinen durch ein erneutes Verschlüsseln von x und einen Vergleich mit dem Verschlüsseltem. Sollte das zugrundeliegende Blockkryptosystem nun sicher sein, kann auch davon ausgegangen werden, dass der daraus entstandene MAC sicher ist.
Dies ist also ein sicherer MAC (insofern ein sicheres Blockkryptosystem verwendet wird), jedoch kann dieser nur einen Block der Länge ℓ mit einem Etikett versehen. Um nun das System für Nachrichten beliebiger Länge zu erweitern, müssen einige Dinge beachtet werden.

Ein zu kurzer Text kann leicht mit Padding auf die gewünschte Länge gebracht und dann normal behandelt werden. Ein Text x, der zu lang ist, wird zuerst in Blöcke kürzerer Länge aufgeteilt. Es wird erst einmal angenommen, der Text würde in Blöcke der Länge ℓ aufgeteilt. Sollte dies nicht genau aufgehen (|x| ist nicht durch ℓ teilbar), wird der letzte Block mit Padding auf die Länge ℓ gebracht. Leider kann nun nicht einfach der Etikettieralgorithmus auf alle Blöcke angewendet werden und die Ergebnisse hintereinander gehängt werden. Würde dies umgesetzt werden, könnte ein Angreifer, der die Nachricht abfängt, leicht Blöcke aus Nachrichten vertauschen, Blöcke weglassen oder die Reihenfolge der Blöcke ändern, bevor er die Nachricht weiterschickt. Dies würde natürlich bedeuten, dass dieses Verfahren unsicher ist und muss verhindert werden.
Eine Möglichkeit, um all dies zu verhindern, ist es, jedem Block 3 Dinge hinzuzufügen: einen Blockzähler i, die gesamte Anzahl der Blöcke a und ein eindeutiger Identifikator der Nachricht n. Durch i kann die Reihenfolge der Blöcke nicht unbemerkt geändert werden, durch a kann kein Block unbemerkt weggelassen werden und durch n können Blöcke nicht unbemerkt mit anderen Nachrichten vertauscht werden. i, a und n werden nun an jeden Textblock x_i angehängt, bevor der Algorithmus E den jeweiligen Block verschlüsselt.
Somit müssen jedoch alle Informationen x_m, i, a und n gemeinsam eine Länge von ℓ aufweisen um von E verschlüsselt werden zu können. Die Aufteilung von x muss also überarbeitet werden, sodass Platz für i, a und n bleibt. Formal ist es das Einfachste, jedem der vier Werte gleich viel Platz einzuräumen. x würde also durch $\frac{\ell}{4}$ geteilt, sodass jedes x_m eine Länge von $\frac{\ell}{4}$ hat. Dabei muss nun gelten $\ell >= 4$. Außerdem kann ein Problem auftreten, wenn ℓ zu klein im Vergleich zur Nachrichtenlänge ist, da dann eventuell nicht genug Platz für die nötigen Informationen in den einzelnen Blöcken ist. So könnte bei z.B. $\ell = 8$ eine Nachricht der Länge 10 ($|x| = 10$) nicht übermittelt werden, da 5 Blöcke benötigt würden ($10/(\frac{8}{4}) = 5$) und in den 2 Bits, die n zur Verfügung stehen, die Zahl 5 nicht binär codiert werden kann. Dadurch entsteht für jedes ℓ eine maximale Nachrichtenlänge von $|x|_\mathrm{max} = 2^{\frac{\ell}{4}} * \frac{\ell}{4}$. Die gleiche Beschränkung wird auch durch a auferlegt.
Dies ist zwar eine relativ sichere MAC-Konstruktion, jedoch ist diese aufgrund der großen Aufteilung von x (in $\ell/4$) sehr ineffizient. Eine bessere, auch auf Blockkryptosystemen basierende Alternative ist der jetzt folgende CBC-MAC.

4.1.3 Der CBC-MAC

Der CBC-MAC ist auch ein MAC, der auf Blockkryptosystemen basiert. Wie der Name bereits vermuten lässt, nutzt diese Konstruktion den CBC-Modus eines Blockkryptosystems. Die formale Definition basierend auf Küsters und Wilkes Lehrbuch ([KW11]) bleibt der aus dem vorherigen Kapitel sehr ähnlich:

Definition 4.3 (CBC-MAC)**.** Sei $B = (\{0,1\}^\ell, K, \{0,1\}^\ell, E, D)$ mit $K \subseteq \{0,1\}^s$ für $\ell, s > 0$ ein ℓ-Blockkryptosystem. Der MAC aus B, sei $M_{\text{CBC}} = (K, T_{\text{CBC}})$
mit $T_{\text{CBC}}(x\colon \{0,1\}^{\ell+}, k\colon K)\colon \{0,1\}^\ell$
1. Zerlege x in ℓ-Blöcke: $x = x_0 ... x_{\text{m-1}}$
2. Setze $y - 1 = v$
3. Für $i = 0, ..., m - 1$ bestimme $y_{\text{i}} = E(y_{\text{i-1}} \oplus x_{\text{i}}, k)$
4. Gib $t = y_{\text{m-1}}$ aus.
Dabei bezeichnet v einen fest gewählten Initialisierungsvektor $v \in \{0,1\}^\ell$.

Wie leicht zu erkennen ist, ist der einzige Unterschied im Etikettieralgorithmus zu finden. Dieser ist dem Verschlüsselungsalgorithmus eines Blockkryptosystem im CBC-Modus sehr ähnlich. So wird in ℓ-lange Teile geteilt, die mit dem vorherigen Ergebnis (y) XOR-Verknüpft und dann verschlüsselt werden. Als erstes Ergebnis ($y_{\text{-1}}$) wird ein fester Intialisierungsvektor genutzt. Der eigentliche MAC / die Prüfsumme ist das allerletzte Ergebnis t.

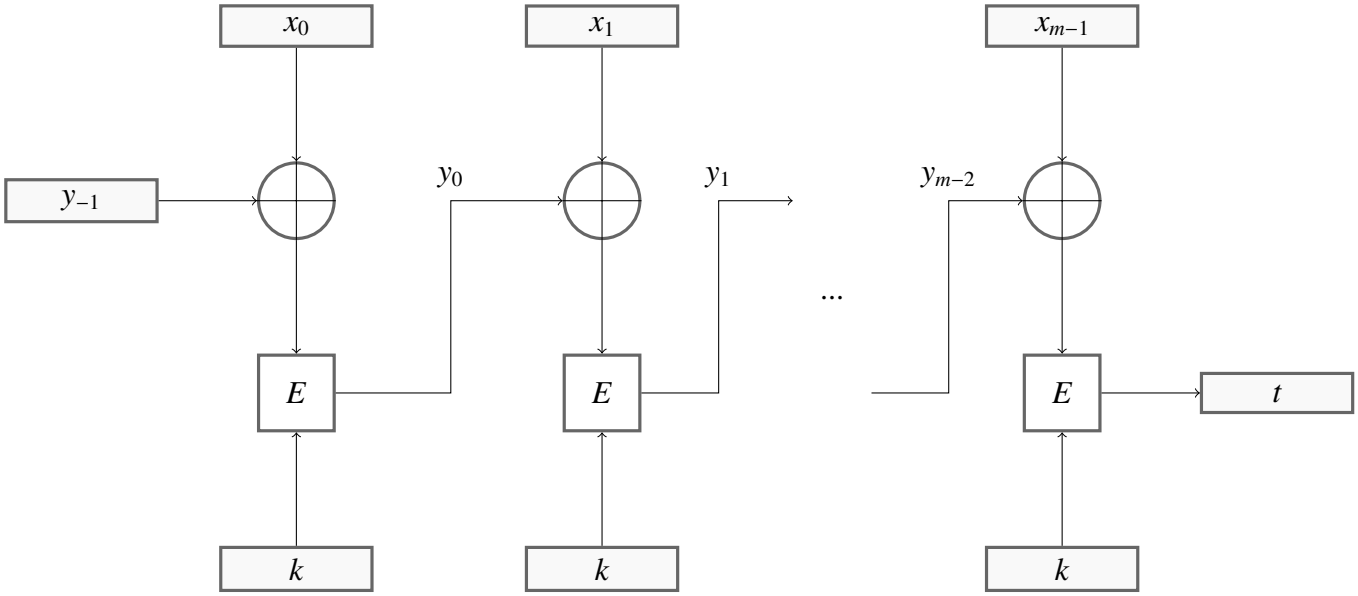

Abbildung 4.1: CBC-MAC

Der CBC-MAC ist eine der ältesten MAC-Konstruktionen. Er ist ein internationaler Standard und ist mit kleinen Variationen weit verbreitet.

Beispiel 4.1 (CBC-MAC)**.** B sei das aus Beispiel 3.1 bekannte ℓ-Blockkryptosystem $B_{10} = (\{0,1\}^\ell, K, \{0,1\}^\ell, E, D)$, für das gilt $\ell, s = 10$ und dessen Ver- und Entschlüsselungsalgorithmen durch $e(x,k) = x \oplus k$ und $d(y,k) = y \oplus k$ für alle $x, y, k \in \{0,1\}^\ell$ gegeben sind. Der CBC-MAC aus B_{10} sei dann $MAC_{CBC-B_{10}} = (K, T_{CBC})$ mit T_{CBC} wie in 4.3 definiert und $v = 1111111111$.
Für die beispielhafte Eingabe $x = 01011111001111010000$ mit $k = 0101010101$ wird wie

folgt verfahren:
Zuerst wird x in die ℓ langen Blöcke $x_0 = 0101111100$ und $x_1 = 1111010000$ aufgeteilt. Dann wird v mit x_0 bitweise XOR-Verknüpft zu 1010000011. Auf diesen Wert und k wird dann die Verschlüsselungsfunktion E angewendet, sodass sich $y_0 = E(x_0 \oplus v, k) = 1010000011 \oplus k = 1111010110$. Daraufhin wird y_0 mit x_1 bitweise XOR-Verknüpft und darauf und auf k wird dann wieder die Verschlüsselungsfunktion E angewendet. Es ergibt sich $y_1 = t = E(0000000110, k) = 0101010011$.

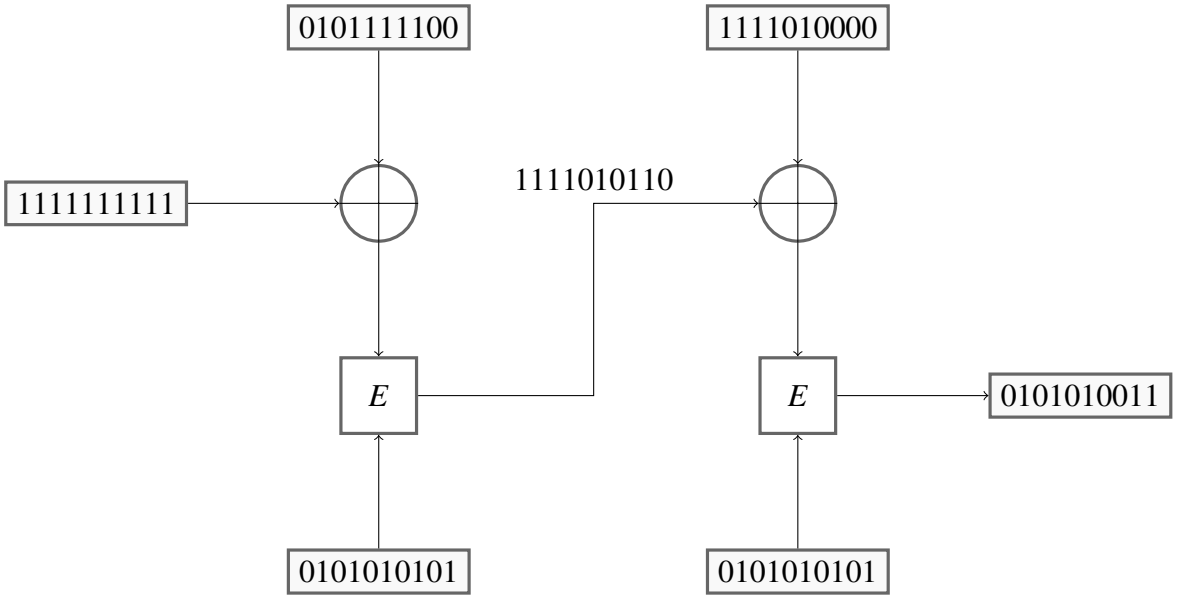

Abbildung 4.2: CBC-MAC mit Werten aus Beispiel 4.1

5 Hashbasierte MACs

In diesem Kapitel wird es um auf Hashfunktionen basierende MACs gehen und somit auch um das Hauptthema dieser Arbeit: den HMAC.
Hashbasierte Authentifizierungsverfahren haben einige Vorteile gegenüber anderen Verfahren. Wie bereits erwähnt, setzen sie auch die Ziele der Integrität und Authentizität um, jedoch sind sie gegenüber anderen MACs darin deutlich schneller bzw. effizienter. So sorgt die Verbindung mit Hashfunktionen dafür, dass Verschlüsselung und somit auch Verifikation im Vergleich zu anderen MACs deutlich schneller ist. Auch bieten die hashbasierten MACs gegenüber anderen MACs durch die Hashfunktionen mehr Sicherheit gegen Angriffe. Auch wenn die hashbasierten MACs im Vergleich zu normalen Hashfunktionen betrachtet werden (sie werden dann als Hashfunktion mit Schlüssel angesehen), sind sie aufgrund des Schlüssels deutlich sicherer. Die genannten Vorteile kommen jedoch auch mit einem eventuellen Nachteil daher: So sind hashbasierte MACs einfach gesagt schneller und sicherer als ihre Konkurrenten, jedoch ist dies nur der Fall, solange der Schlüssel geheim bleibt.
Im Folgenden wird zuerst auf die MAC-Konstruktion der NMACs eingegangen. Danach wird die HMAC-Konstruktion in aller Gründlichkeit erläutert. Ihr Aufbau, ihre momentane Verwendung, Beweise zu ihrer Sicherheit, Angriffe gegen sie und einige interessante Modifikationen ihrer Konstruktion werden betrachtet.

5.1 NMAC

Eine wichtige Konstruktion ist der Nested MAC oder kurz NMAC. Diese Konstruktion geht auf ein Paper von Mihir Bellare, Ran Canetti und Hugo Krawczyk aus dem Jahr 1996 zurück ([BCK79]). In diesem wird die Konstruktion des NMACs vorgestellt :
$NMAC_k(x) = F_{k_1}(F_{k_2}(x))$
x ist hierbei der Klartext und F_k sind jeweils Familien von Hashfunktionen. k_1 und k_2 sind die zwei Schlüssel, die zum Verschlüsseln genutzt werden.
Es wird nun die Füllfunktion betrachtet. Diese sollte MD-kompatibel sein, braucht allerdings noch weitere Eigenschaften um als NMAC-kompatibel zu gelten (Definition nach Küsters und Wilkes Lehrbuch ([KW11])):

Definition 5.1 (NMAC-kompatible Füllfunktion)**.** Eine MD-kompatible (r, D, b)-Füllfunktion p heißt NMAC-kompatible (r, D, b, ℓ)-Füllfunktion, falls $\{0,1\}^\ell \subseteq D$ und $|p(x)| = b$ für alle $x \in \{0,1\}^\ell$ gilt.

Das NMAC Schema kann wie folgt formal definiert werden (Definition nach Küsters und Wilkes Lehrbuch ([KW11])):

Definition 5.2 (NMAC)**.** Es sei f eine (ℓ, b)-Kompressionsfunktion, p eine NMAC-kompatible (r, D, b, ℓ)-Füllfunktion und $H = \{h_k\}_{k\in\{0,1\}^\ell}$ mit $h_k = h_k^{f,p}$ die zugehörige Familie iterierter Hashfunktionen. Der zugehörige NMAC, der mit $NMAC\,[f, p]$ bezeichnet wird, ist das Tupel
$NMAC\,[f, p] = (\{0,1\}^\ell \times \{0,1\}^\ell, T)$
mit $T(x, (k_{out}, k_{in})) = h_{k_{out}}(h_{k_{in}}(x))$ für alle $x \in D, k_{out}, k_{in} \in \{0,1\}^\ell$

Die NMAC-Konstruktion ist zwar nicht unsicher, hat aber ein paar Schwächen, wenn sie zur Anwendung kommen soll. So ist die Benutzung einer Hashfamilie für die Sicherheit der Konstruktion nötig, jedoch ist sie nicht sehr effizient. Auch sind Hashfamilien in Kryptographischen Bibliotheken üblicherweise nicht implementiert, sondern es sind nur einzelne Hashfunktionen enthalten. Auch die Benutzung von zwei Schlüsseln ist nicht so effizient, wie wenn nur ein Schlüssel benutzt wird. All diese Probleme werden mit der HMAC-Konstruktion behoben.

5.2 HMAC

Auch der Hash-based MAC oder HMAC ist eine Konstruktion, die in dem Paper von Mihir Bellare, Ran Canetti und Hugo Krawczyk aus dem Jahr 1996 vorgestellt wurde ([BCK79]). Diese Konstruktion war mehr auf praktische Anwendung ausgerichtet. So nutzt die HMAC-Konstruktion im Gegensatz zum NMAC nur einen Schlüssel und eine Hashfunktion statt einer Familie. Diese Konstruktion hat inzwischen ihren Weg in den Standard gefunden und wird dementsprechend oft benutzt.

5.2.1 Aufbau

Der Aufbau des HMAC ist im Vergleich zu dem des NMAC etwas komplizierter. In dem ursprünglichen Paper ([BCK79]) wird die Konstruktion wie folgt vorgestellt: $HMAC_k(x) = F\left(\bar{k} \oplus opad, F\left(\bar{k} \oplus ipad, x\right)\right)$

x bezeichnet dabei den Klartext und $\bar{k}$ ist der Schlüssel k, der mit Padding auf die Blocklänge b gebracht wurde. $opad$ und $ipad$ stehen für outer und inner Padding, dies sind feste Werte, welche mit dem erweiterten Schlüssel XOR-Verknüpft werden. F stellt die genutzt Hashfunktion dar (siehe auch Abbildung 5.2 im Folgenden).

Dabei werden meist Hashfunktionen basierend auf dem Merkle-Damgård Prinzip verwendet, da diese die gängigsten Hashfunktionen sind und eine gewisse bewiesene Sicherheit aufweisen. Wird nun eine Füllfunktion für die Hashfunktion benutzt, um x auf ein Vielfaches der Blocklänge zu bringen, sollte diese bestimmte Eigenschaften aufweisen (Definition nach Küsters und Wilkes Lehrbuch ([KW11])).

Definition 5.3 (HMAC-kompatible Füllfunktion)**.** Es seien $b, \ell, r > 0$ und sei $D = \{0,1\}^{<2^r - b}$. Eine Funktion $p : \{0,1\}^{<2^r} \to \{0,1\}^{b+}$ heißt HMAC-kompatible (r, b, ℓ)-Füllfunktion, wenn es eine Funktion $p' : \{0, ..., 2^r - 1\} \to \{0,1\}^*$ gibt, sodass $p(x) = x \cdot p'(|x|)$ für jedes $x \in \{0,1\}^{<2^r}$ gilt und die Funktion $\bar{p}(x) = x \cdot p'(b + |x|)$ eine NMAC-kompatible (r, D, b, ℓ)-Füllfunktion ist.

Eine formale Definition des HMAC-Schemas sieht dann wie folgt aus (Definition nach Küsters und Wilkes Lehrbuch ([KW11])):

Definition 5.4 (HMAC)**.** Es sei f eine (ℓ, b)-Kompressionsfunktion, p eine HMAC-kompatible (r, b, ℓ)-Füllfunktion, $u \in \{0,1\}^\ell$ und $m > 0$, sodass $\ell \leq m \leq b$ mit $8|m$ und $8|b$ gilt. Weiter sei $ipad = 00110110$ und $opad = 01011100$. Dann ist das Authentifizierungsschema $HMAC[f, p, u, m]$ definiert durch $HMAC[f, p, u, m] = (\{0,1\}^m, T)$ wobei T durch $T(x, k) = h(k_o \cdot h(k_i \cdot x))$ für alle $x \in \{0,1\}^{<2^r - b}, k \in \{0,1\}^m$ mit $h = h_u^{f,p}, k_i = (k \cdot 0^{b-m}) \oplus ipad^{b/8}, k_o = (k \cdot 0^{b-m}) \oplus opad^{b/8}$ gegeben ist.

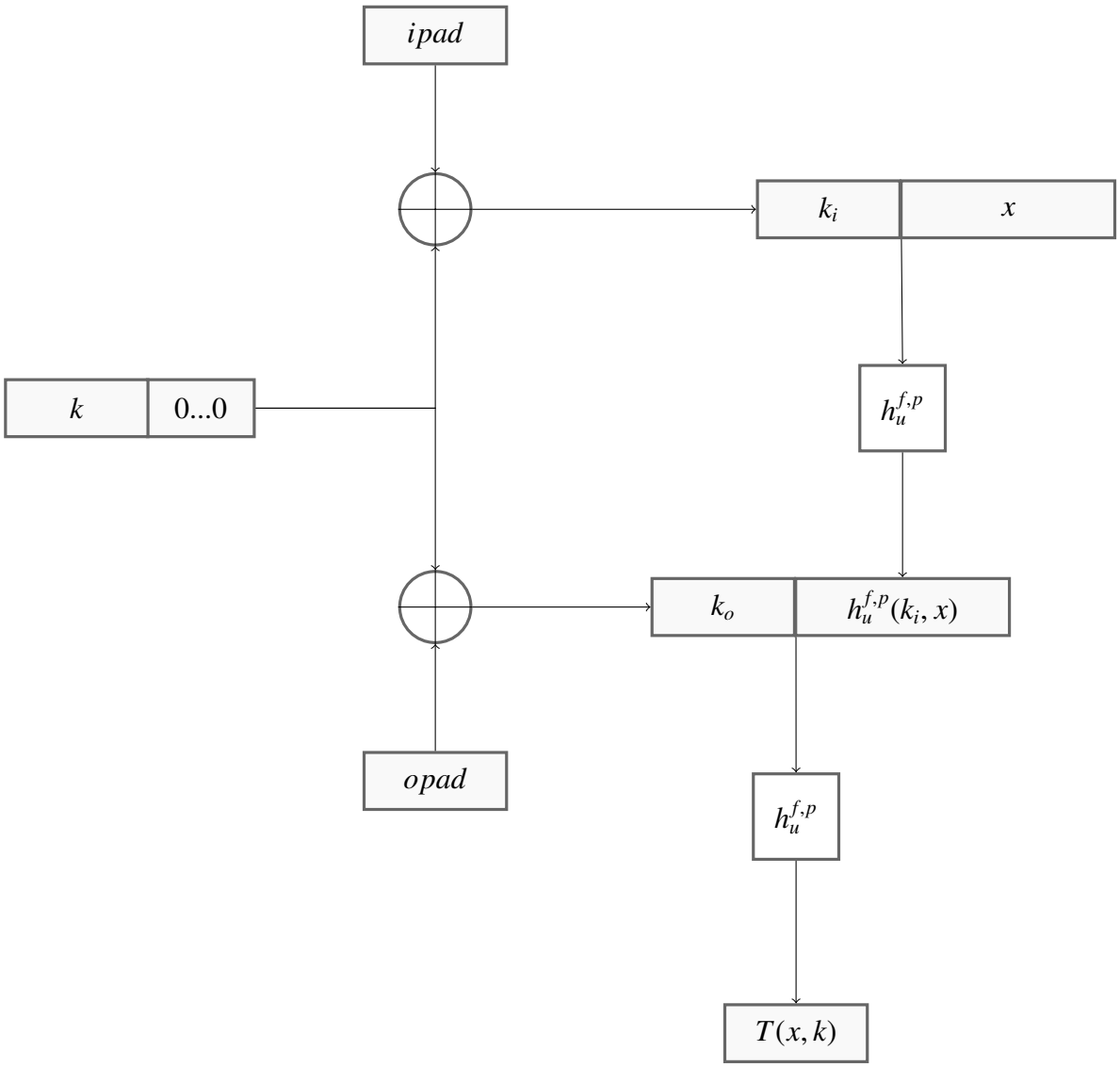

Abbildung 5.1: HMAC-Konstruktion

Beispiel 5.1 (HMAC)**.** Es sei $r = 5$, $\ell = 8$ und $b, m = 16$. Außerdem sei f eine (ℓ, b)-Kompressionsfunktion gegeben durch $f(x, y) = (x \oplus y_{0-7}) \oplus y_{8-15}$ wobei y_{0-7} die Bits 0 bis 7 der Zeichenfolge y bezeichnet und y_{8-15} die Bits 8 bis 15.
p sei die durch $p^{5,16}_{MD}(x) = x \cdot 1 \cdot 0^s \cdot (x)^r_2$ MD-kompatible (r, b)-Füllfunktion, wobei $s \geq 0$ minimal so gewählt wird, dass $b - r = (|x| + 1 + s) \; mod \; b$ gilt. In Beispiel 3.4 wurde schon gezeigt, dass eine solche Füllfunktion MD-kompatibel ist.
Sei nun $D = \{0, 1\}^{<16}$ wie in Definition 5.3 gefordert, sei außerdem $p'(|x|) = 1 \cdot 0^s \cdot (x)^r_2$, wobei s wie oben bei p gewählt wird. Wie leicht zu erkennen ist, ist p' so gewählt, dass wie gefordert $p(x) = x \cdot p'(|x|)$. Damit $p(x)$ eine HMAC-kompatible Funktion ist, muss nach Definition 5.3 nun $\bar{p}(x) = x \cdot p'(b + |x|)$ ein NMAC-kompatible Füllfunktion sein. Damit dies gilt, muss zum einen $\{0, 1\}^\ell \subseteq D$ also $\{0, 1\}^8 \subseteq \{0, 1\}^{<16}$ gelten, was offensichtlich der Fall ist. Außerdem muss $|p(x)| = b$ für alle $x \in \{0, 1\}^\ell$ gelten. Dies ist der Fall, da sich bei einer Zeichenfolge der Länge 8 (ℓ) ein s von 2 ergibt, wodurch sich eine Gesamtlänge von $|p(x)| = 16$ ergibt. Auch für $\bar{p}(x)$ ergibt sich eine Gesamtlänge $\bar{p}(x) = 16$. Das Addieren von b auf den an p' übergebenen Wert beeinflusst die Berechnung von s nicht, da in *mod b* gerechnet wird. Demnach ist $p(x) = p^{5,16}_{MD}(x)$ ein HMAC-kompatible Füllfunktion.
Weiterhin sei $u = 11001100$, $ipad = 00110110$ und $opad = 01011100$.
Dann ist das Authentifizierungsschema $HMAC[f, p^{5,16}_{MD}, 11001100, 16]$ gegeben durch $HMAC[f, p^{5,16}_{MD}, 11001100, 16$
$(\{0, 1\}^{16}, T)$ mit T definiert wie in Definition 5.4.
Für die beispielhafte Eingabe $x = 1001010010$ mit Schlüssel $k = 0101010101010101$ wird wie folgt verfahren:
Zuerst werden die Schlüssel k_i und k_o erzeugt
$k_i = k \oplus ipad^2 = 0101010101010101 \oplus 0011011000110110 = 0110001101100011$ und
$k_o = k \oplus opad^2 = 0101010101010101 \oplus 0101110001011100 = 0000100100001001$.
Auf $k_i \cdot x = 0110001101100011100101001 0 = x_{k_i}$ wird dann die iterierte Hashfunktion angewendet.
So wird x_{k_i} zuerst durch p auf $p(x_{k_i}) = 01100011011000111001010010101100$ erweitert. Danach wird dieser Wert dann in 16 Bit große Blöcke aufgeteilt $x_0 = 0110001101100011$ und $x_1 = 1001010010101100$. Diese werden nacheinander mit der Kompressionsfunktion verschlüsselt, sodass $h(x_{k_i}) = i^f(u, p(x_{k_i})) = i^f(u, x_0 \cdot x_1) = f(f(u, x_0), x_1) = f(11001111, x_1) = 11110111$.
Auf dieses Ergebnis und k_o wird nun erneut die Hashfunktion angewendet, sodass sich $h(k_o \cdot h(x_{k_i}) = 11110111$ ergibt.
$T(x, k) = 11110111$ ist der fertig berechnete MAC/Prüfsumme.

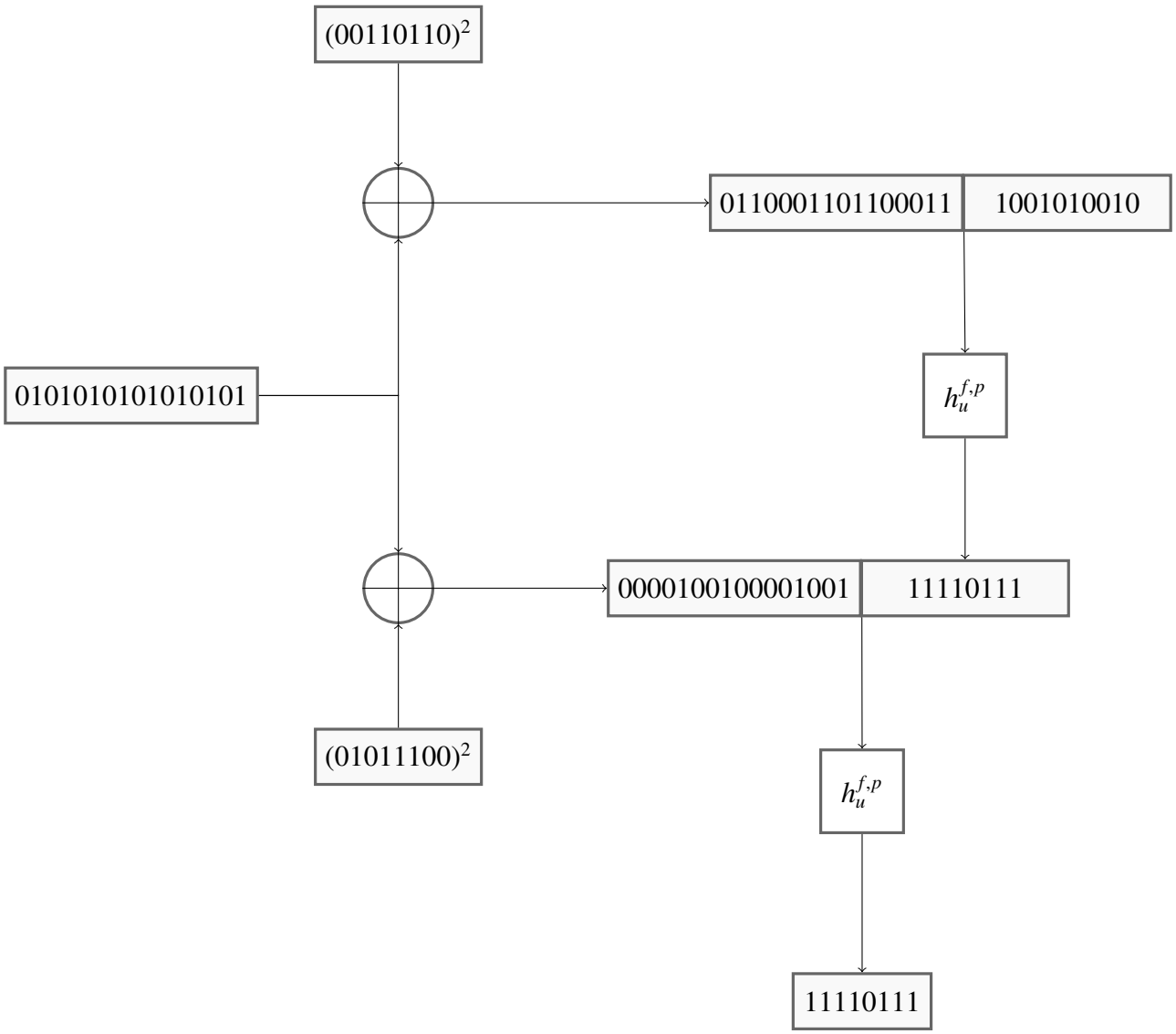

Abbildung 5.2: HMAC-Konstruktion mit Werten aus Beispiel 5.1

5.2.2 Benutzung

Die HMAC-Konstruktion gehört heutzutage zum Standard und wird somit häufig genutzt.

Das National Institute of Standards and Technology (NIST) veröffentlichte bereits 2002 eine Standardisierung für die HMAC-Konstruktion, FIPS 198. Diese wurde im Juli 2008 durch FIPS 198-1 ersetzt ([NIS08]).

Die International Organisation of Standardization hat im Mai 2011 eine Standardisierung zu MACs veröffentlicht, welche auch den HMAC berücksichtigt. Diese wurde im Juni 2021 noch einmal neu aufgelegt in der ISO/IEC 9797-2:2021 ([Sta21]).

Innerhalb des Protokoll suites IPsec (Internet Protocol Security), welcher die sichere Kommunikation im Internet behandelt, wird HMAC genutzt. Es werden hier HMAC-SHA-1 und HMAC-SHA-2, also HMAC Konstruktionen die SHA-1 oder SHA-2 als Hashfunktion nutzen, verwendet ([IET17]).
Auch das TLS (Transport Layer Security) Protokoll nutzt HMAC. Insbesondere werden HMAC-md5, HMAC-SHA-1 und HMAC-SHA256/384 genutzt ([IET18]).

Neben all diesem wird HMAC statt als Authentifizierungsschema oft auch als keyed Hashfunktion, also als Hashfunktion, die einen Schlüssel für erhöhte Sicherheit nutzt, betrachtet.

5.2.3 Sicherheit

Für kryptographische Verfahren ist die Sicherheit das oberste Ziel. Bei symmetrischen Authentifizierungsverfahren und somit bei der HMAC-Konstruktion heißt Sicherheit kurz gesagt, dass MACs nicht fälschbar sein sollen. Die Sicherheit der HMAC-Konstruktion scheint in der Praxis gewährleistet zu sein, dies aber auch zu beweisen beziehungsweise diese Sicherheit auch genau zu bestimmen, ist bis heute noch immer Thema in der Forschung.
Viele Beweise nutzen die enge Verbindung zwischen NMAC und HMAC. Sie zeigen zuerst die Sicherheit von NMAC und nutzen dann aus, dass die Sicherheitsbeweise von NMAC auf HMAC angehoben werden können.

Den ersten Beweis lieferten Mihir Bellare, Ran Canetti und Hugo Krawczyk in eben jenem Paper, das die beiden Konstruktionen vorstellte ([BCK79]). Diese stützen dabei die Sicherheit der Konstruktion auf zwei Bedingungen: Erstens dass die Kompressionsfunktion eine PRF (siehe Definition 3.5) ist und zweitens auf eine schwache Kollisionsresistenz der Hashfunktion. Wie später gezeigt wurde, traf für einige der üblicherweise benutzten Hashfunktionen wie

md5 und Sha-1 die zweite Bedingung nicht zu, wodurch die Sicherheit der Konstruktion nicht mehr bewiesen war.

Es gibt mehrere verschiedene Beweise zur Sicherheit der HMAC-Konstruktion. 2006 erbrachte Mihir Bellare auf der „Advances in Cryptology "(CRYPTO) Konferenz selbst einen neuen Beweis ([Bel06]), welcher lediglich die Bedingung benötigte, dass die Kompressionsfunktion eine PRF ist und stellte so die Sicherheit formal selbst wieder her (auch zu finden in einem Paper aus 2015 ([Bel15])). Auf der CRYPTO Konferenz 2014 wurde der Beweis von Peter Gazi, Krzysztof Pietrzak und Michal Rybár erneut aufgegriffen, vereinfacht und präzisiert ([GPR14]). Zudem gibt es noch viele verschiedene andere Beweise und Paper über die Sicherheit des HMACs.

Auch in Küsters und Willkes Lehrbuch ([KW11]) ist ein Beweis zu finden. Dieser ähnelt sehr dem von Bellare von 2006. Auch bei diesem Beweis wird zuerst die Sicherheit der NMAC-Konstruktion aufgrund der Annahme, dass die Kopressionsfunktion ein PRF ist, bewiesen. Darauf wird dieser Beweis mit gleicher Annahme zur HMAC-Konstruktion angehoben, indem gezeigt wird, dass diese auch als Instanz der NMAC-Konstruktion verstanden werden kann.

5.2.4 Angriffe

Obwohl HMAC als sicher gilt, heißt dies nicht, dass gegen HMAC keine Angriffe existieren. Es gibt mehrere Angriffe gegen HMAC die lediglich genug Zeit, genug Ressourcen oder einen bestimmten Vorteil brauchen, um nutzbar zu sein. Im Folgenden sind einige ausgewählte Angriffe zu finden.

Geburtstagsangriff

Der Geburtstagsangriff macht sich das in Kapitel 3.2 beschriebene Geburtstagsparadoxon zu nutze. So werden bei diesem Angriff zufällige Werte gewählt und die Hashfunktion beziehungsweise der HMAC werden auf sie angewandt. Die entstehenden Ergebnisse werden nun miteinander verglichen, da Gleichheit der Ergebnisse bei unterschiedlichen Eingaben eine Kollision bedeuten würde.
Wie leicht zu erkennen ist, benötigt der Angreifer für diesen Angriff selber die Möglichkeit, die angegriffene HMAC-Konstruktion zu benutzen. Dies ist bei den meisten häufig genutzten HMAC-Implementierungen der Fall, da diese meist standardisierten Entwürfen entsprechen und Standard-Hashfunktionen (so wie md5, Sha 1 oder Sha-2) nutzen, welche für den Angreifer auch zugänglich sind.
Der Geburtstagsangriff benötigt eine Menge Zeit, wenn er mit einer annehmbaren Erfolgswahrscheinlichkeit durchgeführt werden soll. So haben Hashfunktion (und somit auch HMACs)

üblicherweise eine Hashbreite von mindestens 128 Bits. Um nun eine Erfolgswahrscheinlichkeit von über 50% zu haben, müssen nach Küsters und Wilkes Lehrbuch ([KW11]) in etwa 2^{65} Anfragen gestellt und untereinander verglichen werden. Dies ist realistisch betrachtet nicht in absehbarer Zeit machbar.
Wird eine Kollision auf einer Hashfunktion, die auf dem Merkle-Damgård-Prinzip beruht, gefunden, bedeutet dies einen Bruch und sie kann nicht mehr sicher benutzt werden. Ähnliches würde eine Kollision bei einer HMAC-Implementierung bedeuten. Jedoch bedeutet ein Bruch der zugrundeliegenden Hashfunktion einer HMAC-Implementierung nicht zwingend einen Bruch des HMACs. So ist die Hashfunktion md5 zwar gebrochen, HMAC-md5 kann jedoch weiterhin benutzt werden. Dies liegt daran, dass die Konstruktion von HMACs und die Verknüpfung der Hashfunktionen mit den geheimgehaltenen Schlüssel den inneren Zustand des Verfahrens verschleiern und von dem Zustand der alleinstehenden Funktion abgrenzen. Bei einem HMAC sind weiterführende Angriffe nötig, damit eine Kollision der Sicherheit des Verfahrens schadet.

Key-recovery Angriff

Ein Key-recovery Angriff hat das Ziel, den Schlüssel des Verfahrens herauszufinden, um so leicht Nachrichten entschlüsseln und fälschen zu können. Ein solcher Angriff auf HMAC-Verfahren hat einen der beiden erzeugten Schlüssel k_i oder k_o (siehe Def. 5.4) oder auch beide zum Ziel. Der erste Schritt eines solchen Angriffs ist das Finden einer Kollision. Dies kann auf verschiedene Weisen erreicht werden, eine ist ein dem Geburtstagsangriff ähnliches Verfahren. Im Folgenden wird je nach zugrunde liegender Hashfunktion und Ziel des Angriffs der jeweilige Schlüssel aus der Kollision berechnet.

Ein interessanter Key-recovery Angriff wurde von Lei Wang, Kazuo Ohta und Noboru Kunihiro auf der International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT) 2008 vorgestellt ([WOK08]). Der vorgestellte Angriff gilt dem äußeren Schlüssel k_o von HMAC-md4 und nutzt dabei statt einer Kollision eine beinahe Kollision (HMAC-Werte unterscheiden sich um einer zuvor festgelegten Wert), welche schneller zu finden ist. Daraus kann der Angriff nun einen Teil des gesuchten Schlüssels berechnen und mit mehreren Durchläufen kann dieser Angriff auch den gesamten Schlüssel berechnen.
Dieser Angriff ist mit ca. 2^{72} benötigten Vergleichen und ca. 2^{77} offline Berechnungen ein schneller Key-recovery Angriff, liegt aber dennoch weit außerhalb des effizient Berechenbaren.

Es ist heute kein Key-recovery Angriff gegen als sicher geltende HMAC-Konstruktionen bekannt, welcher schnell genug durchgeführt werden kann, um als effizienter Angriff genutzt werden zu können.

Side-channel Angriff

Im Gegensatz zu den anderen hier aufgelisteten Angriffen, greift ein Seitenkanalangriff (Side-channel attack) nicht das kryptographische Verfahren, sondern die physische Implementierung des Verfahrens an. So werden bei diesem Angriff Seitenkanal-Informationen wie die Ausführungszeit, elektromagnetische Ausstrahlungen oder der Energieverbrauch betrachtet und ausgewertet. Aus diesen Informationen kann dann Aufschluss über geheime Informationen innerhalb der Implementierung wie zum Beispiel den geheimen Schlüssel erlangt werden. Aufgrund der verschiedenen Informationen, die betrachtet werden können, gibt es auch verschiedene Klassen von Seitenkanalangriffen.

Eine ist die Differential Power Analysis (DPA), bei welcher der Angreifer den Energieverbrauch des Gerätes, auf dem das angegriffene Verfahren implementiert ist, beobachtet. Dabei wird sowohl der Energieverbrauch während der Ausführung des kryptographischen Verfahrens beobachtet und analysiert, als auch der Verbrauch, während das Verfahren nicht benutzt wird. Zweites dient dem Analysieren von Energieverbrauch, der nicht mit dem Verfahren zusammenhängt. Dieser unwichtige Verbrauch kann so bei der Beobachtung des Verfahrens abgezogen werden. Dies ermöglicht eine sehr viel bessere Betrachtung des Energieverbrauch des eigentlichen Verfahrens. Mit Hilfe eines solchen Angriffs kann der geheime Schlüssel herausgefunden werden, was dem Angreifer das Fälschen der HMACs erlaubt. Einen solchen DPA Angriff auf HMAC-Implementierungen stellte Katsuyuki Okeya auf der Australasian Conference on Information Security and Privacy (ACISP) 2006 vor ([Oke06]), durch welchen der geheime Schlüssel, von HMACs die auf Blockkryptosystemen basieren, herausgefunden werden kann.

Ein anderer Angriff wurde 2009 beim Workshop on Cryptographic Hardware and Embedded Systems von Pierre-Alain Fouque, Gaëtan Leurent, Denis Réal und Frédéric Valette vorgestellt ([Fou+09]). Dieser Angriff setzt die, wie auf dem Workshop gezeigt wurde, in der Praxis häufig vorhandene Bedingung voraus, dass die Hamming-Distanz zwischen bestimmten Registern sichtbar ist. Dieser Angriff kann mit den vorausgesetzten Bedingungen den Schlüssel von bestimmten HMAC-Implementierungen herausfinden und so leicht weitere Nachrichten fälschen.

Wie man sieht, sind die Angriffspunkte bei Seitenkanalangriffen vielfältig und können verheerend für die Sicherheit der HMAC-Implementierung sein. Jedoch greifen, wie bereits erwähnt, diese Angriffe nicht das eigentliche Verfahren an, sondern die Implementierung des Verfahrens. Demnach können diese Angriffe, wenn man von ihrer Existenz weiß, bei der Implementierung beachtet und so leicht verhindert werden, ohne dass das eigentliche Verfahren der HMACs abgeändert werden müsste.

Bedeutung von Angriffen

Die normale HMAC-Konstruktionen sind gegen die genannten Angriffsarten sicher. Entweder wird zu viel Zeit zur Berechnung benötigt, die Erfolgswahrscheinlichkeit ist zu gering oder die Implementierung muss nur mit Beachtung der Sicherheit vorgenommen werden. Doch trotzdem sind die Angriffe und die Forschung in diesem Bereich sehr wichtig. Bei jedem kryptographischen Verfahren besteht immer die Möglichkeit, dass ein neuer Angriff gefunden wird, welcher das Verfahren bricht. Man kann sich nur gezielt gegen bereits bekannte Angriffe schützen. Außerdem zeigt jeder entdeckte Angriff Schwächen auf, welche bei neu entwickelten Konstruktionen verbessert werden können.

5.2.5 Modifikationen

Wie die Angriffe zeigen, ist die HMAC-Konstruktion keinesfalls perfekt und kann noch an einigen Stellen verbessert werden. Aus diesem Grund wurden einige Modifikationen erdacht, welche die ursprüngliche HMAC-Konstruktion verbessern sollen. Im Folgenden werden ein paar dieser Modifikation vorgestellt.

Zwei interessante Modifikationen des HMAC-Schemas stammen von Kan Yasuda. Die erste Modifikation wurde 2007 auf der „International Conference on Cryptology in India“ von ihm vorgestellt ([Yas07]). Diese nennt sich L-Lane HMAC. Diese Konstruktion erzeugt die Prüfsumme auf L Linien mit L Schlüsseln, welche alle aus einem Schlüssel erzeugt werden. Dies geschieht, indem der Klartext in mehrere Teile aufgeteilt wird, welche dann wiederum auf die L Linien verteilt werden. Diese Linien laufen alle wie eine eigene MD-Hashfunktion ab, indem die Kompressionsfunktion nacheinander auf die Teile des Klartextes und das vorherige Ergebnis angewendet wird. Die Ergebnisse der einzelnen Linien werden dann wiederum aneinander gehängt und auf sie wird die Kompressionsfunktion angewendet. Yasuda zeigt auch, dass dieses Verfahren einen HMAC erzeugt, der Sicherheit über das durch das Geburtstagsparadoxon geschaffene Limit hinaus bietet.

Die zweite Modifikation von Kan Yasuda stellte dieser auf der „Information Security Conference“ (ISC) 2009 vor ([Yas09]). Diese simple Modifikation ist der normalen HMAC-Konstruktion sehr ähnlich, ihr einziger Unterschied besteht darin, dass beim zweiten Aufruf der Hashfunktion kein Schlüssel verwendet wird. Die so entstehende Konstruktion nennt Yasuda H^2-MAC und sie kann wie folgt formalisiert werden:
$H^2 - MAC_k(M) = H(H(K \cdot pad \cdot M))$, wobei M der Klartext der Nachricht ist, K der Schlüssel, H die Hashfunktion und pad ein Padding, dessen genaue Definition hier für das Verständnis der Konstruktion nicht relevant ist. Die Verdopplung des Schlüssels K mit $ipad$ und $opad$, wie es in der normalen HMAC-Konstruktion der Fall wäre, entfällt hier, da nur ein Schlüssel benötigt wird.

Genau dies ist auch der Vorteil dieser Konstruktion. So zeigt Yasuda, dass die Sicherheit dieser Konstruktion nahezu gleich wie die der normalen HMAC-Konstruktion ist, sie aber effizienter ist, da nur ein Schlüssel benötigt wird.

Eine weitere interessante Modifikation stammt von Peter Gaži, Krzysztof Pietrzak, und Stefano Tessaro. Diese präsentierten 2015 auf der „Advances in Cryptology (ASIACRYPT)“ Konferenz ([GPT15]) ihre „Whitening HMAC“ Konstruktion, WHMAC. Bei WHMAC besteht der Unterschied darin, dass ein weiterer zufälliger Schlüssel K_w generiert wird, welcher dann mit den einzelnen Textblöcken XOR-Verknüpft wird, bevor diese an die Hashfunktion übergeben werden. Diese Modifikation ermöglicht den Autoren eine sehr genaue Definition der Sicherheit der Konstruktion, wie es bei normalem HMAC noch immer Forschungsgegenstand ist.

Obwohl diese und auch andere Modifikationen Vorteile gegenüber der normalen HMAC-Konstruktion haben, werden sie diese wohl nicht ersetzen. Zum einen kommen die Modifikation auch mit Nachteilen wie erhöhter Berechnungsdauer, der benötigten Abänderung der zugrundeliegenden Hashfunktion oder der Verringerung der Sicherheit gegenüber bestimmten Angriffen. Zum anderen ist dies auch gar nicht der Zweck dieser Modifikationen. Sie dienen eher dazu, die bereits bestehende Konstruktion und die Kryptographie im Bereich der symmetrischen Authentifizierungsverfahren im Allgemeinen besser zu verstehen. Wodurch ein wirklicher Nachfolger dann mit größerem Verständnis und somit höherer Sicherheit entwickelt werden kann, sobald er benötigt wird.

6 Fazit & Ausblick

Ich habe in dieser Arbeit umfassend die Grundlagen zu Authentifizierungsverfahren und Hashfunktionen erklärt. Außerdem habe ich verschiedene MAC-Konstruktionen vorgestellt, so wie den CBC-MAC und den NMAC. So bin ich zur in vielen Punkten anderen MACs überlegenen HMAC-Konstruktion gelangt. Ich habe Funktion und Aufbau der Konstruktion genau erläutert und bin auf ihre Herkunft und Ziele eingegangen. Der Wert der HMAC-Konstruktion sollte klar geworden sein, da diese in wichtigen Internet-Protokollen benutzt wird und auch allgemein als Standard gilt und daher leicht von jedem verwendet werden kann.
Wie die Praxis und auch mehrere Beweise zeigen, ist die HMAC-Konstruktion momentan sicher. Jedoch ist ihre genaue Sicherheit aus mehreren Gründen bis heute Thema der Forschung. Zum einen ist die Sicherheit noch nicht ganz genau bestimmt und zum anderen dient diese Forschung der Zukunft:
So gibt es, wie gezeigt wurde, Angriffe gegen die HMAC zwar momentan nicht angreifbar ist, die aber mit den nötigen Voraussetzungen, wie zum Beispiel einem ausreichend schnellen Computer, dennoch gefährlich sind. Demnach wird die HMAC-Konstruktion, die es bereits seit 1979 als Idee gibt, nicht mehr ewig die nötige Sicherheit bieten und in Zukunft irgendwann ersetzt oder verbessert werden müssen. Deshalb werden zu der Sicherheit und den Angriffen wie gezeigt auch kleine Modifikationen der HMAC-Konstruktion erforscht, um die Konstruktion und die Authentifizierung basierend auf Hashfunktionen genau zu ergründen und zu verstehen. All das erforschte Wissen aus den in dieser Arbeit gezeigten drei Bereichen kann benutzt werden, um einen möglichst guten Nachfolger der HMAC-Konstruktion zu entwickeln, sobald dieser benötigt wird.
Doch für den Moment ist die HMAC-Konstruktion sicher und wird dies wahrscheinlich auch noch eine Weile bleiben. So kann man weiterhin diese MAC-Konstruktion nutzen, um Nachrichten zu authentifizieren und ihre Integrität zu überprüfen.

Momentan werden auch weiterhin die genaue Sicherheit, Angriffe und Modifikationen von HMAC erforscht. Auch gibt es momentan viele Paper zu HMAC-Implementationen in ganz genau bestimmten Zusammenhängen. Ein interessantes Gebiet zur weiteren Betrachtung wären unter anderem das der Digitalen Signaturen, welche in vielen Punkten den HMACs ähnlich sind, aber auch große Unterschiede aufweisen. Auch die Verbindung von HMAC mit der neuartigen Hashfunktion SHA-3 könnte sehr interessant sein, da diese nicht auf dem in dieser Arbeit hauptsächlich betrachteten Merkle-Damgård-Prinzip basiert, sondern eine

sogenannte Schwamm-Konstruktion hat.

Alles in allem gebe ich in dieser Arbeit einen umfassenden Überblick über das Thema der HMACs und erläutere alle nötigen Zusammenhänge, sowie die nötigen Grundlagen und relevante Vorläuferkonstruktionen.

Literatur

[BCK79] M. Bellare, R. Canetti und H. Krawczyk. *Secrecy, authentication, and public key systems*. 1979.

[Bel06] Mihir Bellare. "New Proofs for NMAC and HMAC: Security without collision-resistance". In: *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*. Hrsg. von Cynthia Dwork. Bd. 4117. Lecture Notes in Computer Science. Springer, 2006, S. 602–619. DOI: 10.1007/11818175_36. URL: https://doi.org/10.1007/11818175%5C_36.

[Bel15] Mihir Bellare. "New Proofs for NMAC and HMAC: Security without Collision Resistance". In: *J. Cryptol.* 28.4 (2015), S. 844–878. DOI: 10.1007/s00145-014-9185-x. URL: https://doi.org/10.1007/s00145-014-9185-x.

[Dam90] I.B. Damgård. "A Design Principle for Hash Functions". In: *Advances in Cryptology — CRYPTO' 89 Proceedings. CRYPTO 1989. Lecture Notes in Computer Science, vol 435.* Springer, New York, NY, 1990, S. 416–427. URL: https://link.springer.com/content/pdf/10.1007%2F0-387-34805-0.pdf.

[Fou+09] Pierre-Alain Fouque u. a. "Practical Electromagnetic Template Attack on HMAC". In: *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*. Hrsg. von Christophe Clavier und Kris Gaj. Bd. 5747. Lecture Notes in Computer Science. Springer, 2009, S. 66–80. DOI: 10.1007/978-3-642-04138-9_6. URL: https://doi.org/10.1007/978-3-642-04138-9%5C_6.

[Gol03] Oded Goldreich. *Foundations of cryptography / Oded Goldreich ; Vol. 1: Foundations of cryptography : basic tools*. Cambridge University Press, 2003. ISBN: 9780511546891. URL: https://www.tib.eu/de/suchen/id/TIBKAT%3A883494582.

[GPR14] Peter Gazi, Krzysztof Pietrzak und Michal Rybár. "The Exact PRF-Security of NMAC and HMAC". In: *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*. Hrsg. von Juan A. Garay und Rosario Gennaro. Bd. 8616.

Lecture Notes in Computer Science. Springer, 2014, S. 113–130. URL: https://doi.org/10.1007/978-3-662-44371-2%5C_7.

[GPT15] Peter Gazi, Krzysztof Pietrzak und Stefano Tessaro. “Generic Security of NMAC and HMAC with Input Whitening”. In: *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*. Hrsg. von Tetsu Iwata und Jung Hee Cheon. Bd. 9453. Lecture Notes in Computer Science. Springer, 2015, S. 85–109. DOI: 10.1007/978-3-662-48800-3_4. URL: https://doi.org/10.1007/978-3-662-48800-3%5C_4.

[IET17] Internet Engineering Task Force (IETF). *rfc8221. Cryptographic Algorithm Implementation Requirements and Usage Guidance*. 2017. URL: https://datatracker.ietf.org/doc/html/rfc8221.

[IET18] Internet Engineering Task Force (IETF). *rfc8446. The Transport Layer Security (TLS)*. 2018. URL: https://datatracker.ietf.org/doc/html/rfc8446.

[KW11] Ralf Küsters und Thomas Wilke. *Moderne Kryptographie - Eine Einführung*. Vieweg + Teubner, 2011. ISBN: 978-3-519-00509-4.

[Mer79] R.C. Merkle. *Keying hash functions for message authentication*. 1979. URL: http://www.merkle.com/papers/Thesis1979.pdf.

[NIS08] US Department of Commerce NIST. *FIPS 198-1. The Keyed-Hash Message Authentication Code (HMAC)*. 2008. URL: https://csrc.nist.gov/publications/detail/fips/198/1/final.

[Oke06] Katsuyuki Okeya. “Side Channel Attacks Against HMACs Based on Block-Cipher Based Hash Functions”. In: *Information Security and Privacy, 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006, Proceedings*. Hrsg. von Lynn Margaret Batten und Reihaneh Safavi-Naini. Bd. 4058. Lecture Notes in Computer Science. Springer, 2006, S. 432–443. DOI: 10.1007/11780656_36. URL: https://doi.org/10.1007/11780656%5C_36.

[Rie14] Gerd Riehl. “Alte Geburtstagsprobleme – neu gelöst”. In: *Mathematische Semesterberichte* (2014). DOI: https://doi.org/10.1007/s00591-014-0132-6.

[RS04] P. Rogaway und T. Shrimpton.h. *Cryptographic hash-function basics: Definitions, implications and separations for preimage resistance, second-preimage resistance, and collision resistance.* 2004. URL: www.cs.ucdavis.edu/%CB%9Crogaway.

[Sta21] International Organization for Standardization. *ISO/IEC 9797-2:2021. Information security — Message authentication codes (MACs) — Part 2: Mechanisms using a dedicated hash-function*. 2021. URL: https://www.iso.org/standard/75296.html.

[WOK08] Lei Wang, Kazuo Ohta und Noboru Kunihiro. "New Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5". In: *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*. Hrsg. von Nigel P. Smart. Bd. 4965. Lecture Notes in Computer Science. Springer, 2008, S. 237–253. DOI: 10.1007/978-3-540-78967-3_14. URL: https://doi.org/10.1007/978-3-540-78967-3%5C_14.

[WW14] Christian Wenzel-Benner und Daniel Wasserrab. "Kryptographische Hashfunktionen: Historie, Angriffe und aktuell sichere Standards". In: *Automotive - Safety & Security 2014 (2015), Sicherheit und Zuverlässigkeit für automobile Informationstechnik, Tagung, 21.-22.04.2015, Stuttgart, Germany*. Hrsg. von Herbert Klenk u. a. Bonn: Gesellschaft für Informatik e.V., 2014, S. 79–94. URL: https://dl.gi.de/20.500.12116/2462.

[Yas07] Kan Yasuda. "Multilane HMAC - Security beyond the Birthday Limit". In: *Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings*. Hrsg. von K. Srinathan, C. Pandu Rangan und Moti Yung. Bd. 4859. Lecture Notes in Computer Science. Springer, 2007, S. 18–32. DOI: 10.1007/978-3-540-77026-8_3. URL: https://doi.org/10.1007/978-3-540-77026-8%5C_3.

[Yas09] Kan Yasuda. "HMAC without the SSecond"Key". In: *Information Security, 12th International Conference, ISC 2009, Pisa, Italy, September 7-9, 2009. Proceedings*. Hrsg. von Pierangela Samarati u. a. Bd. 5735. Lecture Notes in Computer Science. Springer, 2009, S. 443–458. DOI: 10.1007/978-3-642-04474-8_35. URL: https://doi.org/10.1007/978-3-642-04474-8%5C_35.