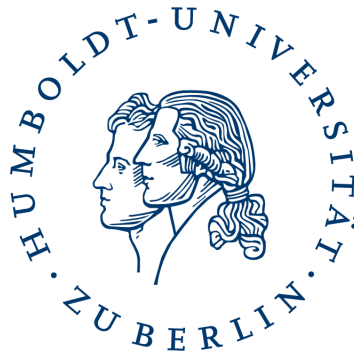


# Humboldt-Universität zu Berlin

Institut für Informatik  
LFE Komplexität und Kryptografie



## MD5 - Aufbau und differentielle Analyse

SE Aktuelle Entwicklungen in der Kryptografie

vorgelegt von  
**Andreas Grüner**  
(Matrikel-Nr. 521149)

16. März 2011

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>2</b>
1.1	Einführung . . . . .	2
1.2	Geschichtlicher Abriss . . . . .	2
<b>2</b>	<b>Aufbau MD5</b>	<b>3</b>
2.1	Notationen . . . . .	3
2.2	Überblick . . . . .	3
2.3	Padding . . . . .	4
2.4	Bufferinitialisierung . . . . .	4
2.5	Hashwertgenerierung . . . . .	4
2.6	Unterschiede zu MD4 . . . . .	6
<b>3</b>	<b>Differentieller Angriff</b>	<b>7</b>
3.1	Differentielle Analyse . . . . .	7
3.2	Klassifikation Angriffe auf Hashverfahren . . . . .	7
3.2.1	Preimage-Attack . . . . .	8
3.2.2	Second-Preimage-Attack . . . . .	8
3.2.3	Collision-Attack . . . . .	8
3.3	Einordnung des differentiellen Angriffs . . . . .	8
3.4	Verfahren des differentiellen Angriffs . . . . .	9
3.4.1	Allgemeines Prinzip . . . . .	9
3.4.2	Two-Block Collision nach Wang et al. [YuWa] . . . . .	10
<b>4</b>	<b>Anwendung - Fälschung digitaler Zertifikate</b>	<b>12</b>
4.1	Einführung digitale Zertifikate . . . . .	12
4.2	Szenario . . . . .	13
4.3	Durchführung . . . . .	13
<b>5</b>	<b>Abbildungs- und Tabellenverzeichnis</b>	<b>16</b>
<b>6</b>	<b>Literaturverzeichnis</b>	<b>17</b>

# 1 Einleitung

## 1.1 Einführung

Hashfunktionen werden in der Informatik zur Sicherstellung der Integrität von Daten genutzt. Eine dieser Hashfunktionen ist Message Digest 5 (MD5). Die Hashfunktion berechnet aus den eingegebenen Informationen einen Prüfwert. Diese Größe wird auf eine sichere Art gespeichert und bei Überprüfungen der Integrität der Daten als Vergleichswert für die erneut berechnete Prüfsumme verwendet. Besteht kein Unterschied zwischen dem ursprünglichen und dem aktuellen Ergebnis der Hashfunktion, kann davon ausgegangen werden, dass die Daten nicht verändert wurden. Im anderen Fall sind die Informationen manipuliert worden. Ist es jedoch möglich zu verschiedenen Datensätzen den selben Hashwert zu konstruieren oder gelingt es ohne viel Aufwand Kollisionen der Hashfunktion zu erzeugen, geht die sicherheitliche Aussagekraft von Prüfwerten dieser Hashfunktion verloren.

In dieser Arbeit wird näher auf das Finden von Kollisionen der Hashfunktion MD5 durch die Anwendung des differentiellen Angriffs eingegangen. Begonnen wird mit einem kurzen geschichtlichen Abriss von MD5. Danach wird der Aufbau von MD5 beschrieben. Im Anschluss wird die differentielle Analyse und der daraus resultierende differentielle Angriff vorgestellt. Das Ende der Arbeit bildet ein Abschnitt über die Fälschung digitaler Zertifikate als Anwendung von Kollisionen in Hashfunktionen.

## 1.2 Geschichtlicher Abriss

Message Digest 5 wurde 1991 von Ronald Rivest am Massachusetts Institute of Technology als neuestes Mitglied der Message Digest Hashfunktionsreihe entwickelt [Riv]. Schon 1993 wurden zwei Nachrichten von Boer und Bosselaers präsentiert, die bei geschickter Wahl des Initialisierungsvektors von MD5 kollidieren. 3 Jahre später, im Jahre 1996, stellte Hans Dobbertin eine Kollision bei gleichem Initialisierungsvektor vor [Pet]. Einen generellen Angriff zur Erzeugung von kollidierenden Nachrichtenpaaren, die bestimmten Bedingungen genüge, und unter Verwendung des originalen, im RFC festgelegten, Initialisierungsvektor von MD5 wurde 2004 von Wang et. al. publiziert [YuWa]. Vastil Klima veröffentlichte 2005 eine verbesserte Version der Angriffsstrategie [Kli05a]. In den darauffolgenden Jahren gab es weitere Publikationen, die sich mit Methoden zur Geschwindigkeitssteigerung der Kollisionsfindung beschäftigten [Kli05b,Kli06,Ste].

## 2 Aufbau MD5

### 2.1 Notationen

In der nachfolgenden Beschreibung der Funktionsweise und des Aufbaus von MD5 werden folgende Notationen und Prämissen verwendet. Bei der Betrachtung eines Bytes befindet sich das Most Significant Bit an der linkensten Position im Byte. Im Hinblick auf die Verwendung eines Wortes tritt das Most Significant Byte an der rechtensten Stelle im Wort auf [Riv].

Die nachfolgende Tabelle präsentiert die genutzten Operatoren.

Symbol	Operator
$\wedge$	bitweises und
$\vee$	bitweises oder
$\neg$	bitweises Komplement
$\oplus$	bitweises xor
$+/-$	Addition/Subtraktion mod $2^{32}$
$\lll s$	Rotation nach links um s Bit
xy	Konkatenation von x und y

Tabelle 1: Operatoren

### 2.2 Überblick

Der Ablauf der Hashwertgenerierung bei MD5 kann grob in die folgenden Schritte eingeteilt werden. Als erstes wird die Eingabenachricht übergeben. Danach erfolgt die Ausführung eines Paddings, welches die Eingabenachricht auf eine gewünschte Größe erweitert. Anschließend werden die Buffer der Hashfunktion initialisiert und die eigentliche Hashwertberechnung beginnt. Als Ergebnis wird ein 128-Bit Hashwert ausgegeben. Ein Überblick über den Ablauf der Hashwertgenerierung stellt folgende Übersicht zur Verfügung [Riv].

1. Eingabe: n-Bit Nachricht  $x_0 \dots x_{n-1} \in \{0, 1\}^n$
2. Padding
3. Bufferinitialisierung
4. Hashwertgenerierung
5. Ausgabe: 128-Bit Hashwert  $y_0 \dots y_{127} \in \{0, 1\}^{128}$

### 2.3 Padding

Während des Verfahrens des Paddings wird an das Ende der eingegebenen Nachricht eine 1 angefügt. Anschließend wird die Eingabe mit einer bestimmten Anzahl von Nullen im Bereich von 0 bis 512 aufgefüllt. Das Padding wird auch ausgeführt wenn die zu hashenden Daten bereits durch 512 teilbar sind [Riv].

Als Ergebnis des Padding liegt eine modifizierte Nachricht  $x_0 \dots x_{n'-1}$  vor die folgende Bedingung erfüllt:

$$n' = n + |t| + 1 \equiv 448 \pmod{512}$$

### 2.4 Bufferinitialisierung

Es werden 4 Wortbuffer bei der Generierung des Hashwertes benutzt. Die Buffer A,B,C und D werden wie folgt initialisiert [Riv]:

Buffer	Initialisierung
A	$01234567_{16}$
B	$89ABCDEF_{16}$
C	$FEDCBA98_{16}$
D	$76543210_{16}$

Tabelle 2: Bufferinitialisierung

### 2.5 Hashwertgenerierung

Die Hashwertgenerierung startet bei der Eingabe der gepaddeten modifizierten Nachricht. Die 4 genutzten wortgroßen Buffer zur Hashwertgenerierung werden mit den im vorangegangenen Abschnitt beschriebenen Werten initialisiert. Als erstes wird die Nachricht in 512-Bit große Blöcke aufgeteilt. Die nachfolgenden Operationen werden iterativ über jeden Nachrichtenblock ausgeführt. Es erfolgt eine Zerlegung der 512-Bit Gruppe in 16 32-Bit Wörter. Des Weiteren werden in 4 wortgroße Hilfsbuffer AA, BB, CC und DD die Werte der Buffer A, B, C und D gesichert. Anschließend erfolgt durch die Ausführung mehrerer Funktionen eine Veränderung der Werte in den Speicherstellen A, B, C und D. Nach dem Abschluß aller Berechnungen werden die Inhalte von A, B, C und D in dieser Reihenfolge konkateniert als Hashwert der eingegebenen Nachricht ausgegeben.

Der nachfolgende Pseudocode veranschaulicht auf einer detaillierten Weise die beschriebene Funktionsweise von MD5 [Köb].

```

input  $x \in \{0, 1\}^{n'}$ 
(A,B,C,D) := (01234567, 89ABCDEF, FEDCBA98, 10325476)
sei  $y = M_1 \dots M_r, r = n'/512$ 
for i := 1 to r do
  sei  $M_i = X[0] \dots X[15]$ 
  (AA,BB,CC,DD) := (A,B,C,D)
  for j := 0 to 63 do
     $(A, B, C, D) := (D, B + (A + f_j(B, C, D) + X[z_j] + y_j) \lll s_j, B, C)$ 
     $(A,B,C,D) := (AA + A, BB + B, CC + C, DD + D)$ 
output ABCD

```

Beschreibung der im MD5 verwendeten Funktion  $f_i$  [Köb]:

$$f_i(X, Y, Z) = \begin{cases} (X \wedge Z) \vee (\neg X \wedge Z), & \text{wenn } j=0, \dots, 15, \\ (X \wedge Z) \vee (Y \wedge \neg Z), & \text{wenn } j=16, \dots, 31, \\ X \oplus Y \oplus Z, & \text{wenn } j=32, \dots, 47, \\ Y \oplus (X \vee \neg Z), & \text{wenn } j=48, \dots, 63. \end{cases}$$

Tabellen zu den verwendeten Konstanten in den Funktionen des MD5.

	$z_j$
$j = 0, \dots, 15$	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
$j = 16, \dots, 31$	1, 6, 11, 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12
$j = 32, \dots, 47$	5, 8, 11, 14, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2
$j = 48, \dots, 63$	0, 7, 14, 5, 12, 3, 10, 1, 8, 15, 6, 13, 4, 11, 2, 9

Tabelle 3: Konstante  $z_j$

	$s_j$
$j = 0, \dots, 15$	7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22
$j = 16, \dots, 31$	5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20
$j = 32, \dots, 47$	4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23
$j = 48, \dots, 63$	6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21

Tabelle 4: Konstante  $s_j$

## 2.6 Unterschiede zu MD4

Die Hashfunktion MD5 basiert auf seinem Vorgänger MD4 und ähnelt dadurch diesem. Zur Erhöhung der Kollisionsresistenz und anderer Sicherheitseigenschaften wurden verschiedene Änderungen an MD5 vorgenommen.

Als erstes ist die Aufstockung der Rundenanzahl von 63 auf 64 Schritte zu nennen. Eine weitere Veränderung bezieht sich auf die genutzten Konstanten  $z_j$  und  $s_j$ . Wurden beim MD4 in jeder Runde dieselbe Konstanten verwendet, ändern sich die Parameter beim MD5 in jedem Schritt. Zusätzlich wurden sie für bessere Sicherheitseigenschaften optimiert. Des Weiteren wurde die Funktion  $f_i$  angepasst [Köb].

## 3 Differentieller Angriff

### 3.1 Differentielle Analyse

Bei der differentiellen Analyse werden Differenzen zwischen verschiedenen Eingabepaaren, deren Hashwerte und den Zwischenresultaten bei der Hashwertberechnung betrachtet.

Eine Differenz eines Eingabepaares ist folgendermaßen definiert:

$$\Delta X = X \text{ op } X' \text{ mit } op \in \{\oplus, -\}$$

Gegeben seien zwei Eingaben  $X, X'$  aus dem Definitionsbereich der Hashfunktion  $H$ . Sowohl  $X$  als auch  $X'$  bestehen aus  $k$  512 Bit Blöcken:  $X = (X_0, \dots, X_{k-1}), X' = (X'_0, \dots, X'_{k-1})$

Als Differential bezeichnet man die Differenz der Hashwerte von  $X$  und  $X'$

$$\Delta H_0 \xrightarrow{(X, X')} \Delta H$$

$H_0$  stellt in diesem Zusammenhang die Differenz der Initialisierungsvektoren der Hashfunktion dar, welche immer 0 beträgt, da die Startwerte vorgegeben sind.  $H$  charakterisiert die Differenz der Hashwerte von  $X$  und  $X'$ . Beträgt der Wert von  $H=0$  besitzen  $X$  und  $X'$  die selbe Prüfsumme und es tritt eine Kollision auf.

Das Differential von  $X$  und  $X'$  kann auf zwei detailliertere Weisen dargestellt werden. Eine Möglichkeit besteht darin die Teildifferenzen nach den einzelnen Runden miteinzubeziehen.

$$\Delta H_0 \xrightarrow{(X_0, X'_0)} \Delta H_1 \xrightarrow{(X_1, X'_1)} \Delta H_2 \xrightarrow{(X_2, X'_2)} \dots \xrightarrow{(X_{k-2}, X'_{k-2})} \Delta H_{k-1} \xrightarrow{(X_{k-1}, X'_{k-1})} \Delta H$$

Zusätzlich können die einzelnen Teildifferenzen der Runden durch die Zwischendifferenzen der  $m$  Schritte verfeinert werden.

$$\Delta H_i \rightarrow \Delta S_{i,0} \rightarrow \dots \rightarrow \Delta S_{i,m-1} \rightarrow \Delta H_{i+1}$$

Im Zuge der granulareren Aufgliederung des Differenzials erhält man eine Folge von Differenzen, die als Differentialpfad bezeichnet werden. Je mehr Differenzen der Differentialpfad umfasst, desto größer ist der Informationsgewinn über das Verhalten der Differenzen während der Hashwertberechnung [YuWa].

### 3.2 Klassifikation Angriffe auf Hashverfahren

Angriffe auf Hashverfahren werden in die 3 folgenden Kategorien eingeteilt: Preimage-Attack, Second-Preimage-Attack und Collision-Attack.

### 3.2.1 Preimage-Attack

Die Preimage-Attack beschreibt einen Angriff auf Hashfunktionen bei dem es möglich ist aus einem gegebenen Hashwert ohne großen Aufwand ein zugehöriges Urbild zu ermitteln. Im Folgenden ist die Attacke formal beschrieben [Köb].

Gegeben: Hashfunktion  $H : X \rightarrow Y$  und  $y \in Y$

Gesucht: Ein Text  $x \in X$  mit  $H(x) = y$

### 3.2.2 Second-Preimage-Attack

Die Second-Preimage-Attack charakterisiert einen Angriff auf Hashfunktionen, bei dem eine Nachricht und der zugehörige Hashwert gegeben ist. Eine weitere Nachricht, die von der ersten Nachricht verschieden ist aber denselben Hashwert besitzt, lässt sich ohne hohen Aufwand berechnen. Die formelle Verdeutlichung steht nachfolgend [Köb].

Gegeben: Eine Hashfkt.  $H : X \rightarrow Y$  und ein  $x \in X$  sowie  $y \in Y$  mit  $H(x) = y$ .

Gesucht: Ein Text  $x' \in X$  mit  $x' \neq x$  und  $H(x') = y$ .

### 3.2.3 Collision-Attack

Bei der Collision-Attack werden zwei beliebige verschiedene Texte im Definitionsbereich einer Hashfunktion gesucht, die den selben Hashwert besitzen. Die formale Definition bestimmt sich Folgendermaßen [Köb].

Gegeben: Eine Hashfkt.  $H : X \rightarrow Y$ .

Gesucht: Texte  $x \neq x' \in X$  mit  $H(x') = H(x)$ .

## 3.3 Einordnung des differentiellen Angriffs

Der differentielle Angriff kann im Bereich der Collision-Attack eingeordnet werden. Als Ausgangspunkt dienen zwei verschiedene zufällig gewählte Texte aus dem Definitionsbereich der Hashfunktion. Im Verlauf des Angriffs werden diese Texte dahingehend modifiziert, dass die finalen Hashwerte kollidieren.

## 3.4 Verfahren des differentiellen Angriffs

### 3.4.1 Allgemeines Prinzip

Die generelle Vorgehensweise des differentiellen Angriffs startet bei der Suche eines geeigneten Differentialpfades der in einer Kollision mündet, d.h. die Differenz der finalen Hashwerte ist 0. Die Elemente der Sequenz repräsentieren Differenzen, die im Zuge der Hashwertberechnung nach dem Ende eines Schrittes oder einer Runde auftreten. Aus dem aufgestellten Differentialpfad werden Bedingungen abgeleitet. Wenn das Differential zweier Nachrichten diesen Bedingungen genügt folgt es dem Differentialpfad und es tritt eine Kollision der Hashwerte auf. Wurden die Prämissen aus dem Differentialpfad extrahiert erfolgt als letzter Schritt eine probabilistische Suche nach einem Eingabepaar von Nachrichten für die Hashfunktion, welche dem Differentialpfad folgen. Werden diese Schritte erfolgreich durchgeführt erhält man als Ergebnis eine Kollision. In den nachfolgenden Abschnitten werden die einzelnen Maßnahmen detaillierter betrachtet anhand von MD5 [Ste].

**Differentialpfad Generierung** Die ersten Differentialpfade wurden per Hand konstruiert. In [Ste] wird eine Methode zur automatischen Generierung von Differentialpfaden vorgeschlagen. Das beschriebene Verfahren basiert auf der Erstellung von zwei getrennten Differentialpfaden, die am Ende des Prozesses miteinander verkettet werden.

Tabelle 5 zeigt ein Teil eines Differentialpfades für MD5 unter Berücksichtigung der Schritte 16, 17 und 18. Die erste Spalte charakterisiert den Schritt innerhalb eines Blockes. In den darauffolgenden Spalten befinden sich die Werte der Differenzen der Hilfsregister A,B,C und D.

Schritt t	$\Delta A$	$\Delta B$	$\Delta C$	$\Delta D$
$t = 16$	$81000000_{16}$	$60000000_{16}$	$7FFF8008_{16}$	$80000000_{16}$
$t = 17$	$80000000_{16}$	$80000000_{16}$	$60000000_{16}$	$7FFF8008_{16}$
$t = 18$	$7FFF8008_{16}$	$80000000_{16}$	$80000000_{16}$	$60000000_{16}$

Tabelle 5: Auszug Differentialpfad MD5

**Extraktion von Bedingungen** Aus dem aufgestellten Differentialpfad können Bedingungen extrahiert werden. Eine Bedingung muss vor oder nach einer Operation gültig sein, damit das Differential der Nachrichten dem Differentialpfad folgt. Diese Klauseln können im Rahmen des differentiellen Angriffs als Bitmuster dargestellt werden. Genügen die Differenzen der Nachrichten dem Bitmuster in allen Schritten tritt eine Kollision bei der Hashwertberechnung auf. Bedingungen werden in die Kategorien hinreichend und zusätzlich eingeteilt, dabei ist die Erfüllung der hinreichenden Konditionen ausreichend für die Konstruktion einer Hashwertkollision [Ste].

In Tabelle 6 werden hinreichende Bedingungen für die Schritte 10,11 und 12 des Registers B dargestellt. In der linken Spalte befindet sich der betrachtete Schritt. Die Rechte Spalte stellt die Bedingungen für das Hilfsregister B dar. Eine 1 oder 0 vermittelt, dass der entsprechende Binärwert an dieser Stelle des Registers stehen muss. Ein Punkt deutet an, dass keine Bedingung für dieses Bit existiert. T bedeutet, dass an dieser Bitstelle der Wert der entsprechenden Position des Registers C vorhanden sein muss.

Schritt t	Hilfsregister B
$t = 10$	0111.... 0..11111 1101...0 01...00
$t = 11$	0010.... ....0001 1100...0 11...10
$t = 12$	000...TT ....1000 0001...1 0.....

Tabelle 6: Auszug hinreichender Bedingungen MD5

**Probalistische Nachrichtensuche** Während der Phase der probalistischen Nachrichtensuche werden Nachrichten ermittelt welche die aufgestellten Bedingungen erfüllen. Dabei wird von einer Anfangsnachricht ausgegangen und diese schrittweise modifiziert damit sie nach und nach alle Bedingungen erfüllt. Diese Methode nennt sich Message Modification und wird in [YuWa] beschrieben.

### 3.4.2 Two-Block Collision nach Wang et al. [YuWa]

Die Strategie der Two-Block Collision nach Wang et al. besteht in dem Gedanken der Erstellung zweier Nachrichten X und X', die aus jeweils zwei Blöcken bestehen. Die Differenz der intermediären Hashwerte nach der Verarbeitung der ersten Blöcke beider Nachrichten soll durch den Unterschied der zweiten Blöcke ausgelöscht werden, damit eine Hashwertkollision der Nachrichten entsteht.

$$X = (X_0, X_1)$$

$$X = (X'_0, X'_1)$$

$$\Delta H_0 \xrightarrow{(X_0, X'_0)} \Delta H_1 \xrightarrow{(X_1, X'_1)} \Delta H$$

Es werden dazu folgende Vorgaben verwendet.

$$\Delta X_0 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, 2^{15}, 0, 0, 2^{31}, 0)$$

$$\Delta X_1 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, -2^{15}, 0, 0, 2^{31}, 0)$$

$$\Delta H_1 = (2^{31}, 2^{31} + 2^{25}, 2^{31} + 2^{25}, 2^{31} + 2^{25})$$

Die einzelnen 32 bittigen Wörter der Differenzen sind in der binary signed digit Darstellung angegeben.

Mit Hilfe dieser Vorgaben und den von Wang et al. entwickelten Differentialpfaden für den ersten und zweiten Block können durch zufälliges Generieren von Nachrichtenpaaren und testen der Bedingungen Kollisionen gefunden werden. Zur Reduzierung des Aufwandes der probabilistischen Suche wurde von Wang et al. die Strategien der Single-Message Modification und der Multi-Message Modification entwickelt [YuWa].

## 4 Anwendung - Fälschung digitaler Zertifikate

### 4.1 Einführung digitale Zertifikate

Bei der Verwendung asymmetrischer Kryptografiealgorithmen existiert das Problem der zweifelsfreien Zuordnung des öffentlichen Schlüssels zu seinem Inhaber. Eine Möglichkeit diese Relation umzusetzen besteht in der Nutzung eines digitalen Zertifikates. Diese Bestätigung verbindet den öffentlichen Schlüssel eines asymmetrischen Schlüsselpaars mit den Personalien des Inhabers. Ein weitverbreiteter Standard ist X.509. Zum Zweck der Überprüfbarkeit der Echtheit des Zertifikates wird dies von einer vertrauenswürdigen Stelle (Certificate Authority) digital signiert. Bei dem Prozess wird von dem öffentlichen Schlüssel und dem restlichen Inhalt des Zertifikats ein Hashwert erstellt. Das Ergebnis der Hashfunktion wird mit Hilfe des privaten Schlüssels der vertrauenswürdigen Instanz verschlüsselt. Möchte eine Person die Authentizität des Zertifikates nachprüfen so muss sie von dem Inhalt erneut den Hashwert berechnen und diesen mit dem verschlüsselten Hashwert vergleichen. Der verschlüsselte Hashwert muss zuvor mit Hilfe des öffentlichen Schlüssels des Signierers entschlüsselt werden. Sind die beiden Hashwerte identisch, ist das Zertifikat unverfälscht. Unterscheiden sie sich dagegen, ist von einer Manipulation auszugehen. Der öffentliche Schlüssel der vertrauenswürdigen Instanz ist dieser wiederum mit einem Zertifikat zugeordnet und von der nächsthöheren Certificate Authority signiert. Dieses System bildet eine Hierarchie und wird Public-Key Infrastruktur genannt. Die Spitze dieser Ordnung bildet die Root Certificate Authority (Root CA) [SoSt].

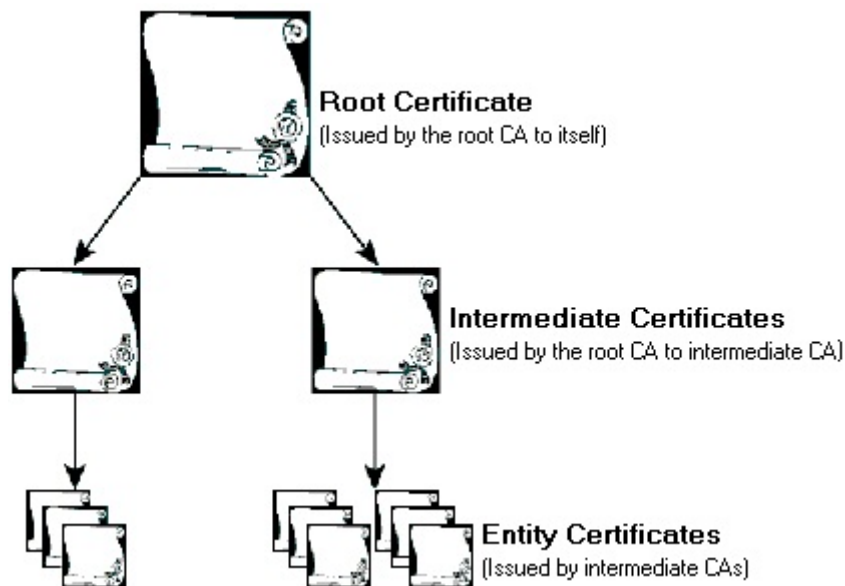


Abbildung 1: Certificate Authority Hierarchie, Quelle: [MsT]

## 4.2 Szenario

Asymmetrische Kryptographie wird oft zur sicheren Verteilung symmetrischer Schlüssel über unsichere Kanäle genutzt. Dies ist beispielsweise der Fall bei dem im Internet genutzten Protokoll SSL/TLS. Hier werden asymmetrische Verschlüsselungsalgorithmen genutzt um symmetrische Sitzungskeys zu verteilen und damit die Kommunikation mit Webservern zu sichern. Der öffentliche Schlüssel des Public-Key Kryptosystems wird mit Hilfe eines Zertifikats verteilt. Das Zertifikat folgt dem X.509 Standard. Wäre es möglich Zertifikate von anerkannten Certificate Authorities signieren zu lassen, welche öffentliche Schlüssel Identitäten zu ordnen, die nicht unter eigener Kontrolle stehen, ist die eindeutige und authentische Zuordnung von öffentlichen Schlüsseln zu Personen oder Institutionen nicht mehr gewährleistet. Des Weiteren ist die Sicherheit zertifikatsbasierter Public-Key Infrastrukturen kompromittiert. Eine andere Möglichkeit besteht darin die Restriktion keine Zertifikate signieren zu dürfen mit Endnutzerzertifikaten zu umgehen. Der X.509 Standard weist ein Feld namens CA auf. Gehört das Zertifikat zu einer Certificate Authority ist der Wert des Feldes True und es ist erlaubt mit dem im Zertifikat enthaltenen Schlüssel andere Zertifikate zu signieren. Dagegen ist bei Endnutzerzertifikaten das eingetragene Datum False. Das vom HashClash Project durchgeführte Szenario besteht in der Erstellung einer gefälschten Certificate Authority. Es erfolgt die Erstellung eines Zertifikates für ein Endnutzers, dass durch eine CA signiert wird, welche als Hashalgorithmus MD5 für die digitale Unterschrift verwendet. Im Anschluss wird versucht ein zweites Zertifikat mit CA=True zu erstellen, dass den Hashwert des Endnutzerzertifikats besitzt. Danach wird die vorhandene Endnutzerzertifikatssignatur unter das neue Zertifikat kopiert. Eine Überprüfung des erstellten Zertifikats wird erfolgreich verlaufen. Mit Hilfe dieser Beglaubigung ist man nun in der Lage selber gültige Zertifikate zu signieren. Mit dieser Fähigkeit lassen sich Zertifikate für Webseiten fälschen. Abbildung 2 gibt einen genaueren Überblick über das Szenario [SoSt].

## 4.3 Durchführung

Zu Beginn der Erstellung des gefälschten CA Zertifikates steht die Beschaffung des normalen Endnutzerzertifikates. Die einzige Bedingung, die die Wahl des Ausstellers einschränkt ist die Benutzung von MD5 als Hashalgorithmus für die digitale Signatur. Die Entscheidung fiel auf RapidSSL. Als nächsten Schritt mussten alle zu signierenden Felder untersucht und ihre Werte bestimmt werden. Es gab folgende zwei Unwägbarkeiten. Einerseits musste der Gültigkeitszeitraum vorhergesagt werden und andererseits war das prognostizieren der Seriennummer unabdingbar. Der Gültigkeitszeitraum beträgt ein Jahr ab dem Ausstellungsdatum. Durch das Ausstellen mehrerer Zertifikate konnten Rückschlüsse auf die Generierung der Seriennummer gewonnen werden. Es erfolgt eine einfache Inkrementierung. Dadurch ist die Seriennummer alleinig von der Anzahl der ausgestellten Zertifikate abhängig. Mit diesen Überlegungen gelang es den Inhalt des zusignierenden Bereiches für einen Zeitpunkt in der Zukunft zu fixieren, an dem die zusignierenden Elemente RapidSSL übergeben werden mussten, damit im Zertifikat die gewünschte Seriennummer und der notwendige Gültigkeitszeitraum eingetragen wird. Man hatte nun

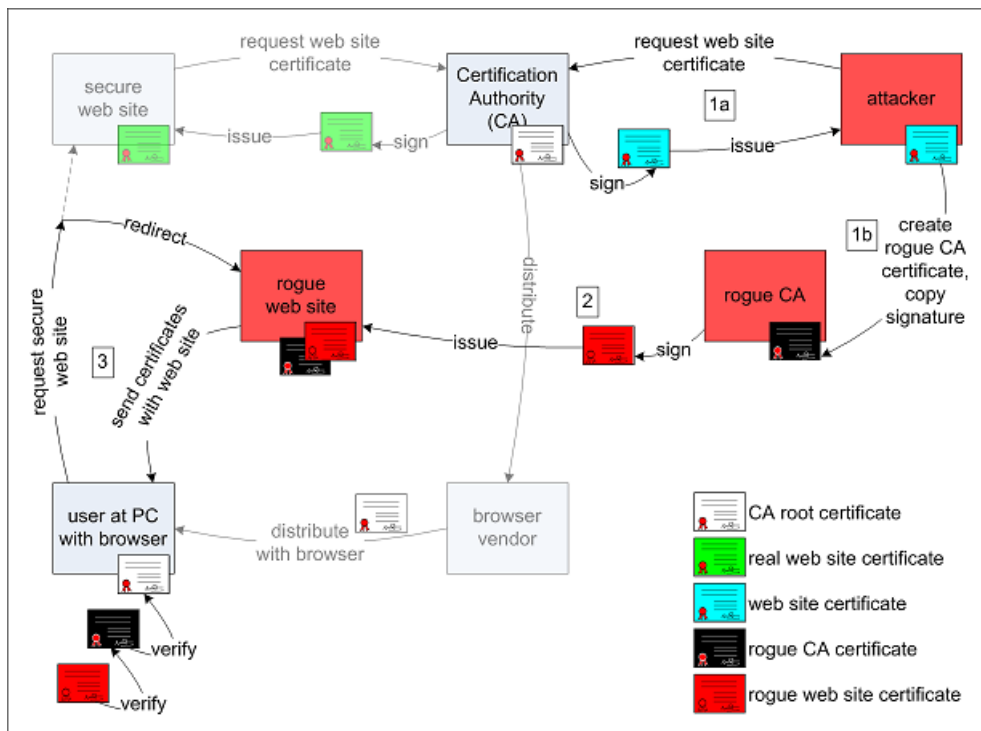


Abbildung 2: Szenario Übersicht, Quelle: [SoSt]

zwei Prototypen des gewünschten Zertifikates. Das Endnutzerzertifikat enthält  $CA=False$  und das andere Zertifikat bezieht  $CA=True$  ein. Diese beiden Nachrichten werden nun als Ausgangspunkt für eine erweiterte Form the Two-Block Collision, genannt Chosen-Prefix Attack, genommen. Eine detaillierte Beschreibung dieses Angriffs befindet sich in [St]. Der variable Bereich der Nachrichten, in dem sie zur Anpassung an die Bedingungen des Differentialpfades modifiziert werden können, ist der Modulus des RSA Schlüssels [SoSt].

## 5 Abbildungs- und Tabellenverzeichnis

Abbildung 1: Certificate Authority Hierarchie. Quelle: [MsT]

Abbildung 2: Szenario Übersicht. Quelle: [SoSt]

Tabelle 1: Operatoren.

Tabelle 2: Bufferinitialisierung. Quelle: [Riv]

Tabelle 3: Konstante  $z_j$ . Quelle: [Köb]

Tabelle 4: Konstante  $s_j$ . Quelle: [Köb]

Tabelle 5: Auszug Differentialpfad MD5. Quelle: [Ste]

Tabelle 6: Auszug hinreichender Bedingungen MD5. Quelle: [Ste]

## 6 Literaturverzeichnis

- [Kli05a] Klima, V.: Finding MD5 Collisions - a Toy For a Notebook. Prag. 2005.
- [Kli05b] Klima, V.: Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications. Prag. 2005.
- [Kli06] Klima, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute. Prag. 2006.
- [Buh] De Buhr, J.: Der MD5-Algorithmus. 2005.
- [YuWa] Wang, X./ Hongbo, Y.: How to Break MD5 and Other Hash Functions. Jinan.
- [Pet] Petrisch, H.: Aktuelle Angriffe auf SHA und MD5.
- [HaPaRo] Hawkes, P./ Paddon, M./ Rose, G.: Musings on the Wang et al. MD5 Collision. Gladenville.
- [Ste] Stevens, M.: On Collisions for MD5. Eindhoven. 2007.
- [WaFeLaHo] Wang, X./ Feng, D./ Lai, X. / Hongbo, Y.: Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. Jinan. 2004.
- [Köb] Köbler, J.: Vorlesungsskript Kryptologie 2. Berlin. 2010.
- [SoSt] Sotirov, A./ Stevens, M./ Appelbaum, J./ Lenstra, A./ Molnar, D./ Osvik, D. A./ De Weger, B.: MD5 considered harmful today. Creating a rogue CA certificate. 2008. URL: <http://www.win.tue.nl/hashclash/rogue-ca/>
- [Riv] Rivest, J.: The MD5 Message-Digest Algorithm. Request for Comments 1321. 1992.
- [MsT] Microsoft Technet: An Introduction to the Windows 2000 Public-Key Infrastructure. URL: <http://technet.microsoft.com/en-us/library/cc768063.aspx>