F A C H H O C H S C H U L E W E D E L

S E M I N A R A R B E I T

Seminararbeit zum Thema:

Post-Quantum-Cryptography

Hash-Based Signature Schemes

Eingereicht von: Antonio Galeazzi

Abgegeben am: 18.11.2017

Dozent: Prof. Dr. Gerd Beuster

Zusammenfassung

Die nachfolgende Arbeit befasst sich mit der Post-Quanten-Kryptographie und dabei insbesondere mit Verfahren zum Signieren von Informationen, die auf Hashfunktionen basieren. Es soll zunächst ein grober Überblick über den derzeitigen Stand der Post-Quanten-Kryptographie gewährt werden, um dann darauf aufbauend Signatur-Verfahren, wie sie heutzutage eingesetzt zu werden, zu betrachten. Davon ausgehend soll gezeigt werden, inwieweit bestehende Signatur-Verfahren durch das Aufkommen von Quanten-Computern bedroht werden könnten, um dann letztlich zu erörtern, auf welche Art und Weise dieser Problematik entgegen gewirkt werden kann.

Inhaltsverzeichnis

Zusammenfassung ..... **II**

Inhaltsverzeichnis ..... **III**

1 Post-Quanten-Kryptographie **1**

2 Digitale Signaturen **2**

2.1 Überblick 2

2.2 Derzeitiger Standard 3

3 Hash-basierte digitale Signaturen **5**

3.1 Hash-Funktionen 5

3.2 Diffie-Lamport Einmal-Signaturen 6

3.2.1 Funktionsweise 6

3.3 Merkle Signatur Schema 8

3.3.1 Funktionsweise 8

4 Sicherheits-Niveau **10**

5 Ausblick **11**

Literaturverzeichnis ..... **IV**

1 Post-Quanten-Kryptographie

Bevor sich der eigentlichen Post-Quanten Kryptographie gewidmet wird, muss diese zunächst von der Quanten-Kryptographie abgegrenzt werden. Bei der Quanten-Kryptographie handelt es sich um kryptographische Verfahren, die auf quantenmechanischen Effekten beruhen. Als bekanntestes Verfahren gilt dabei der sogenannte Quantenschlüsselaustausch.

Davon abzugrenzen ist die Post-Quanten-Kryptographie. Diese setzt sich mit Verfahren auseinander, die auch leistungsfähigen Quanten-Computern standhalten können. Gebiete, die dabei bearbeitet werden, und auch maßgeblich den Inhalt des Buches „Post-Quantum Cryptography“ [1] bestimmen, sind die folgenden:

- „Hash-Based Cryptography“
- „Code-Based Cryptography“
- „Lattice-Based Cryptography“
- „Multivariate-Quadratic-Equations Cryptography“
- „Secret-Key Cryptography“

Diese Arbeit wird sich insbesondere mit „Hash-Based Cryptography“ auseinandersetzen. In diesem Zusammenhang werden vor allem darauf basierende Signatur-Verfahren betrachtet werden.

2 Digitale Signaturen

2.1 Überblick

Bei digitalen Signaturen handelt es sich um einen Mechanismus zum Authentifizieren von Nachrichten. Dabei wird üblicherweise zunächst der Hash einer Nachricht gebildet, der dann mit Hilfe des privaten Schlüssels des Senders verschlüsselt wird. Digitale Signaturen sind mit MACs vergleichbar, bieten jedoch eine Funktionalität, die darüber noch hinausgeht. [2]

Durch die Verwendung von MACs kann sichergestellt werden, dass Nachrichten, die zwei Parteien miteinander austauschen, gegen das Eingreifen eines außenstehenden Dritten geschützt werden. Allerdings werden die zwei miteinander kommunizierenden Parteien alleine dadurch noch nicht gegeneinander geschützt. Dabei sind verschiedene Szenarien denkbar, von denen im Folgenden zwei exemplarisch vorgestellt werden sollen. Dafür wird angenommen, dass Bob eine authentifizierte Nachricht an Alice sendet.

- Alice könnte dann eine andere Nachricht erzeugen und behaupten, dass diese von Bob geschickt worden ist. Dazu müsste Alice lediglich eine neue Nachricht erzeugen und an diese die entsprechende MAC anhängen, die mit Hilfe des Schlüssels, den beide miteinander teilen, erzeugt werden kann.
- Bob kann behaupten, die Nachricht gar nicht gesendet zu haben. Da die Möglichkeit besteht, dass Alice selbst erzeugt haben könnte, kann nicht bewiesen, wer die Nachricht tatsächlich verschickt hat.

In sämtlichen Szenarien, in denen zwischen den miteinander kommunizierenden Parteien kein vollständiges Vertrauen herrscht, ist der Einsatz von MACs nicht ausreichend, um die erforderliche Integrität zu gewährleisten. Dieser Problematik kann durch den Einsatz von digitalen Signaturen entgegengewirkt werden.

Um den Anforderungen gerecht zu werden, müssen digitale Signaturen unter anderem die folgenden Bedingungen erfüllen:

- Die Signatur muss abhängig sein von der Nachricht, die signiert werden soll
- Um einen Missbrauch zu verhindern, muss die Signatur Informationen nutzen, die nur dem Sender bekannt sind
- Es muss verhältnismäßig einfach sein, eine Signatur zu erzeugen
- Es muss verhältnismäßig einfach sein, eine Signatur zu verifizieren
- Es muss praktisch unmöglich sein, für eine bestehende Signatur eine passende Nachricht zu generieren oder zu einer gegebenen Nachricht eine betrügerische Signatur zu erzeugen

2.2 Derzeitiger Standard

Nachfolgend sollen verschiedene Signaturverfahren, die derzeitig in der Praxis verwendet werden, vorgestellt werden. Am Beispiel eines Verfahrens soll das grundsätzliche Vorgehen beim Signieren von Nachrichten verdeutlicht werden.

Heutzutage werden in der Praxis insbesondere RSA, DSA und ECDSA verwendet. [1] Alle diese Verfahren haben gemein, dass ihre Sicherheit im Wesentlichen auf der Schwierigkeit, große Zahlen zu faktorisieren oder den diskreten Logarithmus zu ermitteln, basiert. Diese Eigenschaft wird im weiteren Verlauf von großer Bedeutung sein, wenn Quanten-Computer mitberücksichtigt werden. Von den genannten können Verfahren abgegrenzt werden, die auf Hashfunktionen basieren. Auch diese werden im Zusammenhang mit Quanten-Computern näher betrachtet werden.

Es soll nun anhand des Public-Key-Verfahrens RSA [3] die Funktionsweise von digitalen Signaturen verdeutlicht werden. Bei RSA handelt es sich um ein Public-Key-Verfahren, die sich dadurch auszeichnen, dass es einen öffentlichen Schlüssel gibt, der allen zugänglich ist und zum Verschlüsseln von Nachrichten genutzt wird. Darüber hinaus existieren private Schlüssel, die geheim sind, und zum Entschlüsseln der zuvor mit dem öffentlichen Schlüssel verschlüsselten Nachrichten verwendet wird.

Angenommen, Bob möchte eine private Nachricht an Alice unter Zuhilfenahme eines Public-Key-Verfahrens senden. Dazu verwendet er den öffentlichen Schlüssel E_A von Alice und sendet ihr die verschlüsselte Nachricht $E_A(M)$. Diese Nachricht kann Alice dann durch Verwendung ihres privaten Schlüssels wieder entschlüsseln: $D_A\big(E_A(M)\big) = M$. Eine Besonderheit besteht darin, dass vor dem Versenden der Nachricht kein Austausch von Schlüsseln erforderlich ist. Es muss lediglich garantiert werden, dass potentielle Kommunikationspartner Zugang zum eigenen öffentlichen Schlüssel haben.

Neben der Möglichkeit, RSA zum Verschlüsseln von Nachrichten zu verwenden, kann RSA insbesondere auch herangezogen werden, um Nachrichten mit digitalen Signaturen zu versehen.

Die genannten Eigenschaften gelten für sämtliche Public-Key-Verfahren. Im Folgenden soll die genaue Funktionsweise von RSA betrachtet werden. Die Verschlüsselung und Entschlüsselung folgt bei RSA dem folgenden Muster:

$$C \equiv E(M) \equiv M^e (mod\ n)$$

$$D(C) \equiv C^d (mod\ n)$$

Als „Verschlüsselungs-Schlüssel“, der allen bekannt ist, wird dabei das Paar (e, n) bezeichnet. Das Paar (d, n) bildet den „Entschlüsselungs-Schlüssel“.

Um das Modulo n zu berechnen, ist das Produkt aus p und q, zwei hinreichend großen Primzahlen, zu bilden. Auch wenn n öffentlich ist, ist es nicht möglich dadurch Rückschlüsse auf p und q ziehen zu können. Anschließend ist der private Schlüssel d so zu wählen, dass er teilerfremd zu dem Produkt $(p-1)\cdot(q-1)$ ist. Diese Bedingung kann auch folgendermaßen formuliert werden:

$$\gcd\big(d,(p-1)\cdot(q-1)\big)=1$$

Zuletzt kann e ausgehend von p, q und d als das multiplikative Inverse von d berechnet werden, so dass gilt:

$$e\ \cdot d\ \equiv 1\ (mod\ (p-1)\cdot(q-1))$$

Am folgenden Beispiel soll der Verschlüsselungsmechanismus sowie das Signieren von Nachrichten verdeutlicht werden.

$$\begin{aligned}
&p=7\\
&q=13\\
&n=p\cdot q=7\cdot 13=91\\
&e=5\\
&d\equiv 5^{-1}\equiv 29\ \big(mod\ (7-1)\cdot(13-1)\big)
\end{aligned}$$

Die Nachricht $M=20$ kann dann folgendermaßen verschlüsselt und entschlüsselt werden:

$$\begin{aligned}
&C\equiv M^e\equiv 20^5\equiv 76\ (mod\ 91)\\
&C^d\equiv 76^{29}\equiv 20\equiv M
\end{aligned}$$

Soll diese Nachricht signiert werden, so kann dies durch die folgende Berechnung erfolgen:

$$\begin{aligned}
&S\equiv M^d\equiv 20^{29}\equiv 76\ (mod\ 91)\\
&M'\equiv S^e\equiv 76^5\equiv 20(mod\ 91)
\end{aligned}$$

Üblicherweise würde als Ausgangspunkt einer Signatur nicht die Nachricht M selbst, sondern vielmehr der Hash h(M) der Nachricht, signiert werden, um zu verhindern, dass allein durch die Signatur Rückschlüsse auf die zugrundeliegende Nachricht gemacht werden können.

3 Hash-basierte digitale Signaturen

Von den zuvor genannten Signatur-Verfahren, die auf Public-Key-Systemen beruhen sind, sind jene Verfahren abzugrenzen, deren Sicherheit auf Hashfunktionen basiert. Dies soll im Folgenden an zwei Methoden verdeutlicht werden, die zum jetzigen Zeitpunkt zwar noch keine so verbreitete Anwendung finden, wie die zuvor genannten, aber durch das Aufkommen von Quantencomputern an Bedeutung gewinnen werden.

Zunächst soll das Einmal-Signatur-Verfahren, das von Diffie und Lamport entwickelt worden ist, betrachtet werden, um den Grundstein für das Verständnis von hash-basierten Signaturen zu legen. Darauf aufbauend soll mit dem Merkle-Signatur-Schema ein komplexeres Verfahren in den Vordergrund gerückt werden, das laut vieler Kryptographen [1] ein Ersatz für die derzeitigen Verfahren ist.

Nachdem die Funktionsweisen der beiden Signatur-Verfahren veranschaulicht wurden, soll im Nachgang deren Sicherheit betrachtet werden. Dabei soll sowohl der derzeitige technologische Stand („klassische Computer“) als auch der künftige technologische Stand (Quantencomputer), der unter Umständen möglich wird, berücksichtigt werden.

3.1 Hash-Funktionen

Die Grundlage für hash-basierte Signaturen bilden Hash-Funktionen. [1] Wodurch diese sich auszeichnen soll im Folgenden erläutert werden, um unter anderem zu zeigen, inwieweit sie sich für die Erstellung von digitalen Signaturen anbieten.

Hash-Funktionen akzeptieren Eingaben beliebiger Länge und bilden diese auf einen festen Hash-Wert $H = h(M)$ ab. Dabei zeichnen sich „gute“ Hash-Funktionen insbesondere dadurch aus, dass die Elemente der Zielmenge gleichverteilt als Funktionswerte angenommen werden. Darüber hinaus wird gewünscht, dass auch „minimale“ Änderungen an den Eingabewerten, zu „großen“ Änderungen an den Hash-Werten führen. [2]

Des Weiteren werden die drei folgenden Anforderungen an kryptographische Hash-Funktionen gestellt:

- „Preimage Resistant“: Es ist praktisch unmöglich, zu einem Hashwert $H = h(M)$ das Urbild, also M zu bestimmen.
- „Second Preimage Resistant“: Gegeben sind eine Nachricht M und der dazugehörige Hash-Wert $H = h(M)$. Dann ist es praktisch unmöglich, eine Nachricht M' zu konstruieren, so dass $h(M) = h(M')$ gilt.
- „Collision Resistant“: Es ist praktisch unmöglich zwei beliebige Nachrichten M_1 und M_2 zu finden, so dass $h(M_1) = h(M_2)$ gilt.

Werden alle der genannten Anforderungen erfüllt, so kann von einer „starken Hash-Funktion“ gesprochen werden. [2]

3.2 Diffie-Lamport Einmal-Signaturen

Am Beispiel der Diffie-Lamport Einmal-Signatur soll sich nun Signatur-Schemata gewidmet werden, deren Sicherheit ausschließlich auf der Kollisionsresistenz der zugrundeliegenden kryptographischen Hashfunktion beruht. Dieses und vergleichbare Verfahren gelten als aussichtsreiche Kandidaten für die Post-Quanten-Ära. [1]

3.2.1 Funktionsweise

Zunächst soll die Funktionsweise der Diffie-Lamport Einmal-Signatur, im Folgenden mit LD-OTS (engl.: Lamport-Diffie one-time signature scheme) abgekürzt, erläutert werden. Zunächst werden für das genannte Verfahren zwei Funktionen der folgenden Form benötigt:

$$f : \{0,1\}^n \rightarrow \{0,1\}^n$$
$$g : \{0,1\}^* \rightarrow \{0,1\}^n$$

Dabei handelt es sich bei f um eine Einweg-Funktion und bei g um eine kryptographische Hash-Funktion.

Das Verfahren lässt sich in drei Schritte aufteilen. Zunächst gilt es ein Schlüsselpaar bestehend aus einem Signatur-Schlüssel X und einem Verifikationsschlüssel Y zu generieren. Mit Hilfe des Signatur-Schlüssels X kann ein Dokument dann signiert werden. Zuletzt kann ein zuvor mit X signiertes Dokument mit Y verifiziert werden.

Generation des Schlüsselpaars

Der Signatur-Schlüssel X, der bei LD-OTS verwendet wird, besteht aus zwei n Bit-Strings, deren Länge jeweils n beträgt und die zufällig ausgewählt werden.

$$X = (x_{n-1}[0], x_{n-1}[1], \dots, x_1[0], x_1[1], x_0[0], x_0[1]) \in \{0,1\}^{(n,2n)}$$

Mit $n = 256$ würden demnach 512 Zufallszahlen, die jeweils 256 Bit lang sind, generiert werden. Anschließend kann der Verifikations-Schlüssel Y mit Hilfe der zuvor definierten Einweg-Funktion f berechnet werden.

$$Y = (y_{n-1}[0], y_{n-1}[1], \dots, y_1[0], y_1[1], y_0[0], y_0[1]) \in \{0,1\}^{(n,2n)}$$
$$y_i[j] = f(x_i[j]), 0 \ \leq i \ \leq n, j = 0, 1$$

Um die Schlüssel zu generieren sind demzufolge 2n Berechnungen von f erforderlich. Beide generierten Schlüssel bestehen schließlich aus 2n Zahlen, die jeweils n Bit lang sind.

Generation der Signatur

Ein Dokument $M \in \{0, 1\}^*$ kann mit Hilfe des zuvor generierten Signatur-Schlüssels X signiert werden. Dazu ist zunächst der Hash des Dokuments berechnen:

$$g(M) = d = (d_{n-1}, \dots, d_0)$$

Für die Signatur gilt dann:

$$\sigma = (x_{n-1}[d_{n-1}], \dots, x_1[d_1], x_0[d_0]) \in \ \{0, 1\}^{(n,n)}$$

Verifikation

Um eine zuvor generierte Signatur zu verifizieren ist die Einweg-Funktion f auf die Signatur anzuwenden und mit dem Verifikations-Schlüssel Y zu vergleichen. Wenn beides miteinander übereinstimmt, kann daraus geschlossen werden, dass die Signatur gültig gewesen ist. Formal muss das Folgende gelten:

$$\big(f(\sigma_{n-1}), \dots, f(\sigma_0)\big) = (y_{n-1}[d_{n-1}], \dots, y_0[d_0])$$

Das gesamte Verfahren soll nun an dem folgenden Beispiel verdeutlicht werden. Es gelte:

$$\begin{aligned} &n = 3 \\ &f : \{0,1\}^3 \rightarrow \{0,1\}^3, x \mapsto x + 1 \; mod \; 8 \\ &d = (1,0,1) \end{aligned}$$

Für den Signatur-Schlüssel X gelte:

$$X = (x_2[0], x_2[1], x_1[0], x_1[1], x_0[0], x_0[1]) = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \in \{0,1\}^{(3,6)}$$

Aus diesem kann dann der zugehörige Verifikations-Schlüssel Y berechnet werden:

$$Y = (y_2[0], y_2[1], y_1[0], y_1[1], y_0[0], y_0[1]) = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \in \{0,1\}^{(3,6)}$$

Die Signatur von $d = (1,0,1)$ ist dann:

$$\sigma = (\sigma_2, \sigma_1, \sigma_0) = (x_2[1], x_1[0], x_0[1]) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \in \{0,1\}^{(3,3)}$$

Die Problematik bei LD-OTS besteht darin, dass ein Signatur-Schlüssel nur ein einziges Mal verwendet werden darf. [1] Aus diesem Grund eignet sich das gezeigte Verfahren in dieser Form nur eingeschränkt für die Praxis. Das folgende Beispiel soll veranschaulichen, aus welchem Grund die mehrmalige Verwendung des gleichen Signatur-Schlüssels zu vermeiden ist. Es gelte $n = 4$. Es werden nun zwei Nachrichten mit den Hashwerten $d_1 = (1,0,1,1)$ und $d_2 = (1,1,1,0)$ mit dem gleichen Signatur-Schlüssel signiert. Für die Signaturen gilt dann: $\sigma_1 = (x_3[1], x_2[0], x_1[1], x_0[1]$ und $\sigma_2 = (x_3[1], x_2[1], x_1[1], x_0[0]$. Dadurch erhält ein Angreifer Kenntnis über $x_3[1], x_2[0], x_2[1], x_1[1], x_0[1], x_0[1]$ und könnte dadurch beispielsweise gültige Signaturen für Nachrichten mit den Hashwerten $d_3 = (1,0,1,0)$ und $d_4 = (1,1,1,1)$ erzeugen. Die Sicherheit des Verfahrens beruht auf der praktischen Unmöglichkeit, zu gegebenen Hashwerten die Urbilder zu ermitteln. Je häufiger ein Signatur-Schlüssel X verwendet wird, desto mehr Bestandteile des Schlüssels werden bekannt gegeben, wodurch die Wahrscheinlichkeit steigt, Nachrichten zu generieren deren Hash gültig signiert werden kann.

3.3 Merkle Signatur Schema

Die zuvor vorgestellte Lamport-Diffie Einmal-Signatur ist für den Großteil praktischer Anwendungsfälle aufgrund der Tatsache, dass jedes Schlüsselpaar nur einmal verwendet werden kann, nicht anwendbar. [1] Eine Verbesserung des Verfahrens wird mit dem Merkle-Signatur-Schema (MSS) möglich, das binäre Hash-Bäume verwendet.

3.3.1 Funktionsweise

MSS benötigt eine kryptographische Hash-Funktion sowie ein Einmal-Signatur-Schema. Als Einmal-Signatur-Schema kann beispielsweise die zuvor gezeigte Lamport-Diffie Einmal-Signatur verwendet werden. Auch MSS kann in drei verschiedene Abschnitte aufgeteilt werden. Zunächst sind die Schlüsselpaare zu erzeugen. Anschließend ist eine entsprechende Signatur zu generieren, die dann schließlich im letzten Schritt verifiziert werden kann.

Generation der Schlüsselpaare

Es ist zunächst die Höhe des Hash-Baumes auszuwählen: $H \in \mathbb{N}, H \geq 2$. Die Wahl von H ist ausschlaggebend für die Anzahl der Dokumente, die später signiert werden können. Dabei gilt, dass das generierte Schlüsselpaar genutzt werden kann, um 2^H Dokumente zu signieren. Es werden dann 2^H Schlüsselpaare $(X_j, Y_j), 0 \leq j < 2^H$ generiert. X_j bezeichnet dabei den Signatur-Schlüssel und Y_j den Verifikations-Schlüssel. Die Blätter des Merkle-Baumes werden dann durch die Hash-Werte $g(Y_j), 0 \leq j < 2^H$ der Verifikations-Schlüssel repräsentiert. Die Werte der inneren Knoten werden als Hash-Werte der Konkatenation der Kinder des Knotens gebildet. Letztlich ergibt sich dadurch die Wurzel des Baums als öffentlicher Schlüssel. Der private Schlüssel wird durch die 2^H generierten Signatur-Schlüssel gebildet.

Formal gilt für die Knoten des Baumes:

$$v_h[j] = g(v_{h-1}[2j] \parallel v_{h-1}[2j+1]), 1 \leq h \leq H, 0 \leq j < 2^{H-h}$$

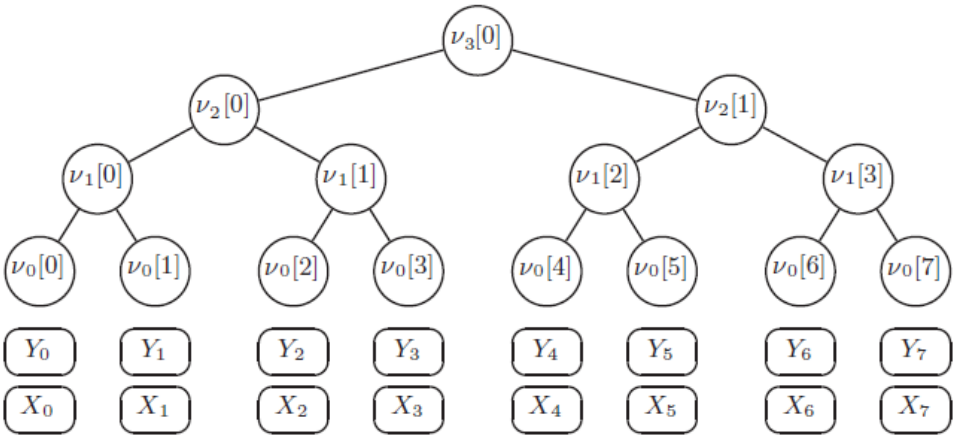

Abbildung 1: Ein Merkle-Baum der Höhe H = 3

Generation der Signatur

Die zuvor generierten Schlüsselpaare können nacheinander nun zum Signieren von 2^H Dokumenten verwendet werden. Wie auch zuvor beim Lamport-Diffie-Verfahren wird zunächst der Hash-Wert $d = g(M)$ berechnet. Die Signatur σ_{OTS} wird dann unter Zuhilfenahme des Einmal-Signatur-Verfahrens OTS erstellt. Dazu kann beispielsweise die bereits vorgestellte Lamport-Diffie Einmal-Signatur verwendet werden. Zur Signierung wird ein privater Schlüssel $X_s, s \in \{0, \dots, 2^H - 1\}$ verwendet. Die Signatur besteht zu diesem Zeitpunkt, wie auch bei der Lamport-Diffie Einmal-Signatur, aus der Signatur selbst sowie dem zugehörigen Verifikationsschlüssel Y_s. Um dem Empfänger zu ermöglichen, die Authentizität des Verifikationsschlüssels Y_s sicherzustellen, wird der Signatur der Index s des verwendeten Schlüsselpaars sowie eine Folge von Knoten $A_s = (a_0, \dots, a_{H-1})$ hinzugefügt. Mit diesen zusätzlichen Informationen ist es möglich, ausgehend vom Verifikationsschlüssel Y_s bzw. des korrespondieren Hash-Wertes $g(Y_s)$ die Wurzel des Merkle-Baums zu berechnen.

Die vollständige Signatur entspricht dann der folgenden Form:

$$\sigma_s = \left(s, \sigma_{OTS}, Y_s, (a_0, \dots, a_{H-1})\right)$$

Das grundsätzliche Vorgehen bei der Erstellung einer Signatur unter Zuhilfenahme eines Merkle-Baums geht aus der nachfolgenden Grafik hervor:

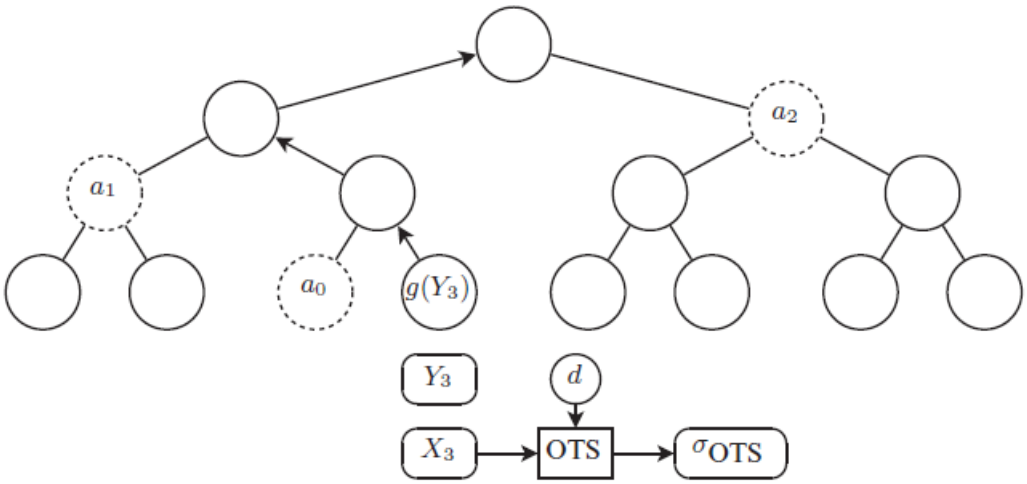

Abbildung 2: Generation einer Merkle-Signatur für s = 3. Die markierten Knoten bilden den Authentifizierungs-Pfad, der der Signatur hinzugefügt wird.

Verifikation

Die Verifikation einer zuvor erstellten Signatur kann in zwei Schritte aufgeteilt werden. Zunächst wird die eigentliche Signatur σ_{OTS} selbst verifiziert. Falls als Einmal-Signatur-Schema Lamport-Diffie verwendet worden ist, ist dieser Schritt der gleiche, der bereits zuvor bei der Betrachtung der Lamport-Diffie Einmal-Signatur gezeigt worden ist. Anschließend ist die Authentizität des Verifikationsschlüssels Y_s zu bestimmen. Dazu verwendet ein Empfänger den Verifikationsschlüssel Y_s und bildet dazu zunächst den Hash-Wert $g(Y_s)$, der dann als Ausgangspunkt genutzt wird, um unter Zuhilfenahme des Indexes s und des Authentifizierungs-Pfads $(a_0, \dots, a_{H-1})$ die Wurzel des Baumes zu berechnen. Falls der errechnete Wert dem Public-Key entspricht, kann der Verifikationsschlüssel Y_s als gültig betrachtet werden.

4 Sicherheits-Niveau

Abschließend soll nun das Sicherheits-Niveau der Merkle-Signatur in Kombination mit der Lamport-Diffie Einmal-Signatur betrachtet werden. Dazu soll sowohl die Sicherheit bei Verwendung von klassischen Computern als auch bei Quanten-Computern berücksichtigt werden.

Die genannten Verfahren können nur attackiert werden, indem Angriffe auf die „preimage resistance“ und die „collision resistance“ der verwendeten Hash-Funktion vorgenommen werden. Diese Attacken basieren auf einer Brute-Force-Methode und machen sich das „birthday“-Problem zu Nutzen. [1]

Klassische Computer

Es kann gezeigt werden, dass das Sicherheits-Niveau b in Abhängigkeit von der Länge der Hash-Werte der folgenden Gesetzmäßigkeit entspricht:

$$b = \frac{n}{2} - 1$$

Dieser Zusammenhang gilt, solange die Höhe des Merkle-Baums $H \leq \frac{n}{3}$ ist und die Länge der Funktionswerte der Hash-Funktion $n \geq 87$ist. Der entsprechende Beweis kann unter [1] nachvollzogen werden.

Quanten-Computer

Ein ähnlicher Zusammenhang kann auch für das Sicherheits-Niveau b bei Betrachtung von Quanten-Computern hergestellt werden. Es gilt:

$$b = \frac{n}{3} - 1$$

In diesem Fall gilt für die Höhe des Merkle-Baums $H \leq \frac{n}{4}$ und für die Länge der Funktionswerte der Hash-Funktion $n \geq 196$. Auch dieser Beweis ist unter [1] zu finden.

Sicherheits-Niveau für Längen der Ausgabewerte

Aus der nachfolgenden Tabelle können exemplarische Werte für die Anzahl der Sicherheits-Bits bei Verwendung verschiedener Hash-Funktionen entnommen werden:

Output length n	128	160	224	256	384	512
Classic case						
bit security b	63	79	111	127	191	255
Maximum value for H	42	53	74	85	128	170
Quantum case						
bit security b	–	–	73	84	127	169
Maximum value for H	–	–	56	64	96	128

Abbildung 3: Sicherheits-Bits bei Verwendung verschiedener Hash-Funktionen

5 Ausblick

Es lassen sich abschließend einige Erkenntnisse festhalten. So lässt sich sagen, dass sämtliche Signatur-Verfahren, die derzeit größtenteils verwendet werden und im Wesentlichen auf den mathematischen Problemen, große Zahlen zu faktorisieren oder den diskreten Logarithmus zu berechnen, basieren, nicht mehr sicher verwendet werden können, sobald Quanten-Computer Realität geworden sind. Die erfreuliche Nachricht ist jedoch, dass schon zum jetzigen Zeitpunkt Verfahren existieren, deren Sicherheit auf kollisionsfreien Hashfunktionen basieren, die selbst Quanten-Computern standhalten können. Dabei ist erstaunlich, dass Quanten-Computer zurzeit zwar unter Umständen noch etwas futuristisch anmuten und zum jetzigen Zeitpunkt auch noch keine direkte Bedrohung darstellen, hash-basierte Signatur-Verfahren, die den Quanten-Computern standhalten können, aber zum Teil auf das Jahr 1979 zurückzuführen sind. Die Verwendung der Lamport-Diffie Einmal-Signatur in Kombination mit Merkle-Bäumen stellt eine vielversprechende zukünftige Lösung dar, die jedoch zum jetzigen Zeitpunkt noch nicht ausgereift ist, da die Performanz aufgrund der Größe der Signatur und der damit verbundenen Speicheranforderung, insbesondere verglichen mit anderen Verfahren wie beispielsweise RSA, sehr groß ist. Auf diesem Gebiet wird derzeit noch intensiv geforscht und es bleibt abzuwarten, welche Neuerungen und Verbesserungen bis zum Erscheinen von leistungsfähigen Quanten.-Computern entwickelt werden können.

Literaturverzeichnis

[1] D. Bernstein, J. Buchmann und E. Dahmen, Post-Quantum Cryptography, Berlin Heidelberg: Springer, 2008.

[2] W. Stallings, Cryptography and Network Security: Principles and Practice, Upper Saddle River: Prentice-Hall, 1998.

[3] R. L. Rivest, A. Shamir und L. Adleman, „A Method for Obtaining Digital Signatures and Public-Key Cryptography," *Magazine Communications of the ACM,* pp. 120-126, 1977.

[4] L. Lamport, „Constructing Digital Signatures from a One Way Function," *SRI International,* 1979.

[5] A. Hülsing, S.-L. Gazdag, D. Butin und J. Buchmann, „Hash-based Signatures: An Outline for a New Standard," *Workshop on Cybersecurity in a Post-Quantum World,* 2015.

[6] J. Buchmann, L. C. C. García, E. Dahmen, M. Döring und E. Klintsevich, „CMSS - An Improved Merkle Signature Scheme," *Proceedings of the 7th international conference on Cryptology in India,* pp. 349-363, 2006.