

Vorstellung der Hashfunktion *Shabal* als Kandidat für den *SHA-3* Standard

Stephan Müller

8. Oktober 2010

Einleitung

Allgemeines

Dies ist eine Ausarbeitung eines von mir gehaltenen Vortrages zum *Shabal*-Algorithmus. Der Vortrag wurde im Juni 2010 im Rahmen eines Kryptographieseminars von Herrn Prof. Dr. Köbler an der HU-Berlin gehalten. Die Ausarbeitung ist als ausführlichere Variante der Vortragsfolien zu verstehen. Sie enthält im Wesentlichen das im Vortrag Gesagte und dient als dessen Ersatz. Diese Ausarbeitung sowie die Vortragsfolien sollten sich auf den Webseiten des Instituts für Informatik der HU-Berlin wiederfinden.

Übersicht

Der Vortrag und diese Arbeit teilen sich in drei Teile. Nach einer kurzen Übersicht über die SHA-3 Standardisierung folgt mit der ausführlichen Beschreibung des *Shabal*-Algorithmus der wichtigste Abschnitt. Dabei habe ich versucht sowohl das Konzept als auch die technischen Details verständlich darzustellen. Bis auf wenige Nebensächlichkeiten wäre nach dieser Beschreibung eine Implementation des Algorithmus möglich. Der dritte Teil behandelt die Sicherheit von *Shabal*. Nach einem erheblichen theoretischen Vorbau wird skizziert, wie die Autoren von *Shabal* die Sicherheit ihres Algorithmus einstufen und begründen. Der dritte Abschnitt ist keineswegs vollständig, er soll nur die Methoden und Ideen zur Bewertung der Sicherheit beleuchten. Die Dokumentation von *Shabal* [3] ist mit 300 Seiten sehr ausführlich und beinhaltet unter anderem detailliert die Anwendung von Teil drei auf *Shabal*.

Inhaltsverzeichnis

1	Der SHA-3 Standard	2
1.1	Shabal	2
2	Beschreibung des Algorithmus	3
2.1	Notation und Vereinbarungen	3
2.1.1	Übersicht - Variablen	3
2.2	Mode of Operation	3
2.2.1	Nachrichtenrunden	3
2.2.2	Verkettung von Nachrichtenrunden	4
2.2.3	Nachrichtenpadding und Initialisierung	5
2.3	Keyed Permutation \mathcal{P}	5
2.3.1	Initiale Runden	6
2.3.2	Hauptunden	6
2.3.3	Finale Transformation	7
2.3.4	Zusammenfassung	7

3 Sicherheitsaspekte	7
3.1 Grundlegende Sicherheitsbegriffe	8
3.1.1 Sicherheitsziele	8
3.1.2 Ideale Kryptosysteme	8
3.1.3 Kryptosysteme	9
3.2 Vergleich kryptographischer Systeme	9
3.2.1 Ununterscheidbarkeit von Kryptosystemen	9
3.2.2 Undifferenzierbarkeit	11
3.3 Anwendungen auf Shabal	11
3.3.1 Undifferenzierbarkeit von einem Zufallsorakel	11
3.3.2 Security Claims	12

1 Der SHA-3 Standard

Das amerikanische Standardisierungsinstitut NIST - *National Institute of Standards and Technology* - leitet den Standardisierungsprozess der neuen Hashfamilie *SHA-3*. Auf der Webseite¹ heißt es

NIST has opened a public competition to develop a new cryptographic hash algorithm, which converts a variable length message into a short 'message digest' that can be used for digital signatures, message authentication and other applications. The competition is NIST's response to recent advances in the cryptanalysis of hash functions. The new hash algorithm will be called 'SHA-3' and will augment the hash algorithms currently specified in FIPS 180-2, Secure Hash Standard.

Die Ausschreibung läuft seit Ende 2007 und befindet sich mittlerweile in der zweiten Auswahlrunde. 14 von ursprünglich 64 Kandidaten sind noch im Wettbewerb. Unter anderem auf den ECRYPT-Seiten² kann man sich einen (inoffiziellen) Überblick über den Verlauf beschaffen. Im August 2010 ist die zweite Konferenz zum Wettbewerb und die Auswahl der finalen Kandidaten geplant. Im Laufe des Jahres 2012 soll nach über sechs Jahren der SHA-3 Standard ausgearbeitet sein.

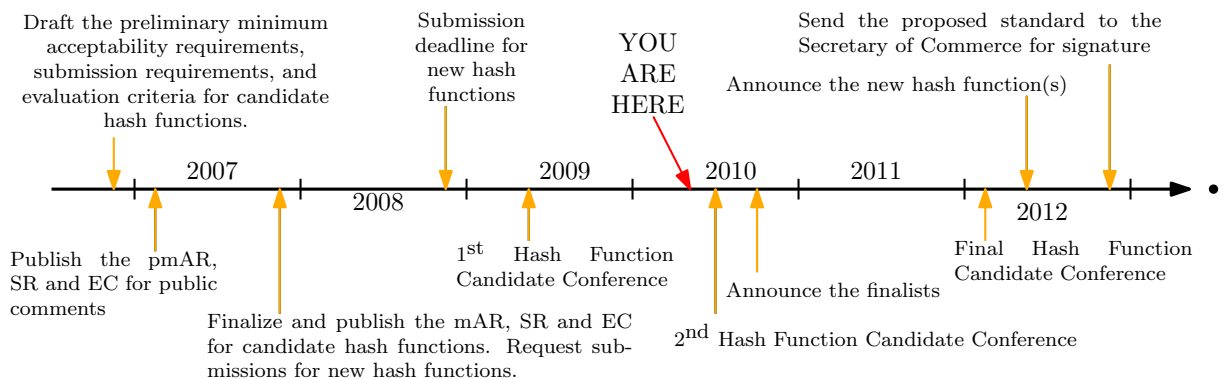


Abbildung 1: Der SHA-3 Standardisierungsprozess im Überblick, Stand: Mai 2010

1.1 Shabal

Einer der verbliebenen 14 Kandidaten ist Shabal. Er wurde vom Gemeinschaftsprojekt *Saphir* (Security and Analysis of Hash Primitives) unter der Leitung der französischen Telekom eingereicht. Der Name Shabal leitet sich von dem französischen Rugbyspieler Sébastien Chabal ab. Während der Sportler durch aggressives Aussehen und Spiel bekannt ist, wirkt der Algorithmus in seinen Ideen eher elegant. Insbesondere der *mode of operation* ist innovativ und unterscheidet sich in seinem Design stark von den weiteren Algorithmen der zweiten Runde. Die Geschwindigkeit von Shabal ist gut, jedoch ist der Platzverbrauch relativ groß. Insbesondere ist er unabhängig von der gewünschten Größe des Hashwertes. Bemerkenswert ist die sehr ausführliche Dokumentation [3] in der vor allem die Designentscheidungen und Sicherheitsbetrachtungen detailliert beschrieben sind.

¹<http://csrc.nist.gov/groups/ST/hash/sha-3/index.html>, 30.Mai 2010

²http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo

2 Beschreibung des Algorithmus

Die Beschreibung hier folgt wesentlich derer im Wettbewerbsbeitrag [3]. Dort werden jedoch drei Sicherheitsparameter (r , s und p) verwendet, die im Folgenden auf die empfohlenen Werte gesetzt und nicht weiter als Parameter betrachtet werden.

Shabal ist stark an das klassische Merkle-Damgård Design angelehnt. Die Nachricht wird blockweise gelesen und stets in identischen Nachrichtenrunden verarbeitet. Die neuen Ideen, die den Algorithmus auszeichnen, sind sein *mode of operation*, das heißt welche Informationen auf welche Weise in die nächste Nachrichtenrunde übermittelt werden, und eine *keyed permutation*, die das Herzstück einer Nachrichtenrunde ist. Jede Nachrichtenrunde erhält drei Eingaben aus der vorherigen Runde: A , B und C . Der aktuelle Nachrichtenblock wird an zwei Stellen durch wortweise Addition bzw. Subtraktion in B und C eingebracht. Gleichzeitig dient er zusammen mit dem unveränderten C als Schlüssel zur Auswahl einer Permutation der Bits von (A, B) .

2.1 Notation und Vereinbarungen

Mit M wird stets die betrachtete Nachricht bezeichnet. Nach einem gewissen Padding kann sie (von links nach rechts) in k Blöcke der Länge 512 bit aufgeteilt werden. Die einzelnen Blöcke werden mit M_ω bezeichnet. Je nach Kontext wird ein Nachrichtenblock als Bitfolge der Länge 512 oder als 16 Worte à 32 bit aufgefasst. Die einzelnen Worte werden mit hochgestelltem Index beginnend bei 0 bezeichnet. Es ist also

$$\begin{aligned} M &= (M_1, M_2, \dots, M_\omega, \dots, M_k) \\ M_\omega &= (M_\omega^0, M_\omega^1, \dots, M_\omega^{15}) \quad 1 \leq \omega \leq k \end{aligned}$$

Mit l_h wird die gewünschte Bitlänge des Hashwertes bezeichnet. Mögliche Werte sind dabei: 192, 224, 256, 384 und 512. Im Folgenden wird hauptsächlich $l_h = 512$ bit betrachtet. \oplus bezeichnet wie üblich (bitweises) XOR. Die eckigen Zeichen \boxplus und \boxminus bezeichnen wortweise Addition bzw. Subtraktion modulo 2^{32} . Insbesondere gibt es keine Überträge von einem Wort auf das nächste. Die *endianness* ist wie im MD5-Verfahren gemischt. Auf Bitlevel wird *big endian* und auf Bytelevel *little endian* benutzt. \ll und \lll bezeichnen Bitverschiebungen nach links. Rechts werden Nullen bzw. in der dreifachen Variante die Bits von links wieder eingefügt. Weitere Notationen entsprechen ihren Standardbedeutungen.

2.1.1 Übersicht - Variablen

Bezeichnung	Bedeutung	Größe
M	Nachricht	$\leq 2^{64}$ bit
k	Anzahl der Blöcke von M	
ω	Nachrichtenblockzähler	64 bit, $1 \leq \omega \leq k$
M_ω	Nachrichtenblock ω	512 bit bzw. 16 Worte à 32 bit
M_ω^i	i -tes Wort eines Nachrichtenblocks $0 \leq i \leq 15$	32 bit
$A = (A_0, \dots, A_{11})$	interne Variable	384 bit bzw. 12 Worte à 32 bit
$B = (B_0, \dots, B_{15})$	interne Variable	512 bit bzw. 16 Worte à 32 bit
$C = (C_0, \dots, C_{15})$	interne Variable	512 bit bzw. 16 Worte à 32 bit
(o_1, o_2, o_3)	Konstanten	$= (13, 9, 6)$
l_h	Länge des Hashwertes	192, 224, 256, 384 oder 512 bit

2.2 Mode of Operation

2.2.1 Nachrichtenrunden

Abbildung 2 stellt eine einzelne Nachrichtenrunde schematisch dar, sie erhält fünf Eingaben $M_\omega, \omega, A, B, C$ und liefert drei Ausgaben A', B' und C' jeweils mit der selben Größe wie die ungestrichenen Eingaben. \mathcal{P} bezeichnet die schon erwähnte schlüsselabhängige Permutation. Die $384 + 512 = 896$ links eingehenden Bits werden in Abhängigkeit von C und M_ω permutiert und wieder in 384 bit und 512 bit große Blöcke aufgeteilt. Die Details hierzu werden im Abschnitt über die Permutation \mathcal{P} erläutert. Der Index des aktuellen Nachrichtenblocks ω wird auf die ersten zwei Worte von A addiert. Der gesamte Nachrichtenblock M_ω

geht mehrfach in die Berechnungen ein, er wird wortweise auf B addiert, dient als Schlüssel zur Auswahl der Permutation und wird wortweise von C subtrahiert.

Man beachte die direkte Konsequenz von $B' = C \oplus M_\omega$. Unabhängig vom Zähler ω erzeugt eine Nachrichtenrunde für verschiedene Nachrichtenblöcke M_ω und M'_ω stets verschiedene Ausgaben in B' . Das bedeutet: möchte man unter Kontrolle von M und sogar A, B und C Kollisionen erzeugen, ist dies nicht innerhalb einer einzelnen Nachrichtenrunde möglich. Man benötigt mehrere Nachrichtenrunden und dann ist gute Kontrolle über die Permutation \mathcal{P} nötig.

2.2.2 Verkettung von Nachrichtenrunden

Die Verkettung von Nachrichtenrunden, in der Einleitung als *mode of operation* bezeichnet ist nach vorhergehender Darstellung der Nachrichtenrunden sehr einfach. Die Eingaben A, B und C erhält eine Nachrichtenrunde aus den Ausgaben A', B' und C' der vorhergehenden Runde wie in Abbildung 3. Besonderheiten treten im mode of operation nur am Anfang und am Ende auf. Es muss geklärt werden, woher die Werte für A, B und C kommen, die in der ersten Nachrichtenrunde benötigt werden. Dieses Problem kann auf mehrere Arten gelöst werden und wird im nächsten Abschnitt behandelt. Zweitens muss geklärt werden, wie nun der gewünschte Hashwert $\mathcal{H}(M)$ abgeleitet wird.

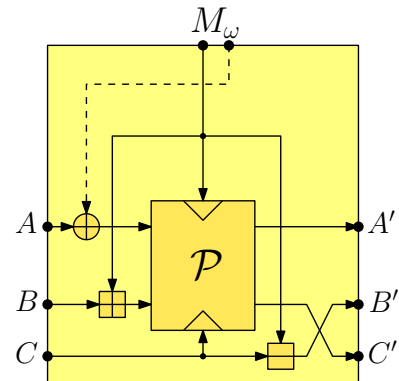


Abbildung 2: schematische Darstellung einer Nachrichtenrunde

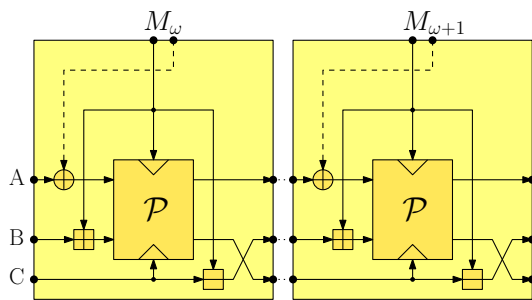


Abbildung 3: Verkettung von Nachrichtenrunden

Der mode of operation wird in zwei Phasen aufgeteilt. Phase Eins bilden die schon bekannten Nachrichtenrunden. Es gibt genau³ k dieser Runden, wobei k wie vereinbart die Anzahl der Blöcke der bereits gepaddeten Nachricht beschreibt. Zusätzlich, als zweite Phase, gibt es noch drei finale Runden. Diese unterscheiden sich nur dadurch von den Nachrichtenrunden, dass der Zähler ω konstant auf k gesetzt wird. Abbildung 4 fasst die Situation zusammen. Die letzte Ausgabe C' beinhaltet den Hashwert $\mathcal{H}(M)$. Ist l_h kleiner als 512 bit so wird ein Teil der Ausgabe verworfen.

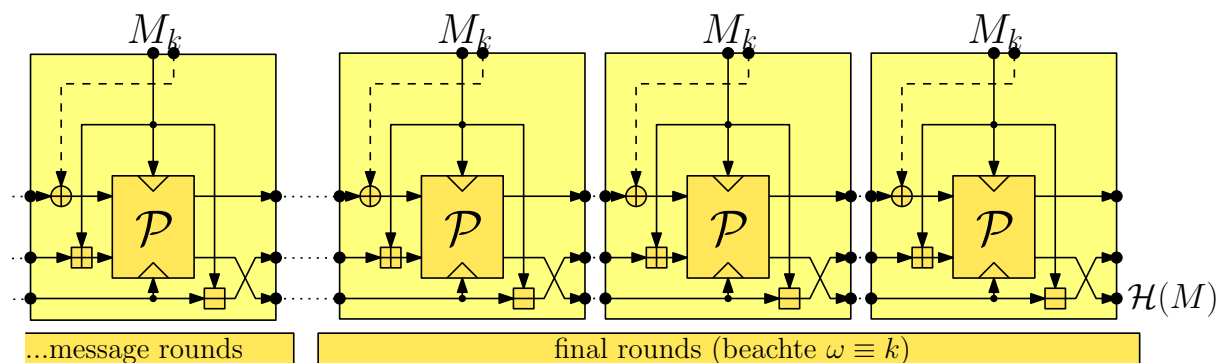


Abbildung 4: Verkettung von Nachrichtenrunden, am Ende werden drei Finalrunden mit $\omega \equiv k$ ausgeführt, C enthält den gewünschten Hashwert $\mathcal{H}(M)$.

³man beachte den Abschnitt 2.2.3 - Prepadding

2.2.3 Nachrichtenpadding und Initialisierung

Nachrichtenpadding Wie bei vielen kryptographischen Algorithmen muss auch bei Shabal die Nachricht vor der Verarbeitung auf entsprechende Länge ergänzt werden. Im Fall von Shabal wird ein klassisches Verfahren benutzt. Der Nachricht wird ein Bitstring 10^n mit $0 \leq n \leq 511$ angehängt, sodass die Nachrichtenlänge (in bit) anschließend ein Vielfaches von 512 ist. Das bedeutet, die Nachricht wird in jedem Fall um mindestens eine 1 ergänzt, auch wenn sie vorher bereits passende Länge gehabt hätte.

Zur Initialisierung der benötigten Variablen A, B und C gibt es zwei Möglichkeiten.

Prepadding In dieser Variante werden der Nachricht zwei Nachrichtenblöcke vorangestellt, welche wortweise die Werte l_h bis $l_h + 31$ enthalten.

$$\begin{aligned} M_{-1} &= (l_h, l_h + 1, \dots, l_h + 15) \\ M_0 &= (l_h + 16, \dots, l_h + 31) \end{aligned}$$

Der Zähler wird auf $\omega = -1$ gesetzt und A, B und C werden mit 0 initialisiert. Der Hashwert kann nun wie oben beschrieben berechnet werden. Es werden dabei dann zwei zusätzliche Nachrichtenrunden für $\omega = -1$ und $\omega = 0$ durchlaufen.

Initialisierung Die Inhalte von A', B' und C' nach den Runden $\omega = -1, \omega = 0$ hängen nicht von der eigentlichen Nachricht ab, sie können also im Voraus berechnet und adäquat abgespeichert werden. A, B, C für die Nachrichtenrunde $\omega = 1$ können dann mit diesen Werten initialisiert werden.

Man beachte, dass die Initialisierungen von l_h abhängen, es ist also nicht möglich aus dem 512 bit Hashwert die Hashwerte für kleinere Werte von l_h abzulesen. Dies ist die einzige Stelle, an der sich die Algorithmen für verschiedene Werte von l_h unterscheiden.

2.3 Keyed Permutation \mathcal{P}

Die schlüsselabhängige Permutation \mathcal{P} ist das Herzstück des Algorithmus. Ihre Beschreibung ist aufwändig und technisch, obwohl die Idee sehr einfach ist. Die Funktion \mathcal{P} erhält vier Eingaben wie in Abbildung 2. Diese werden im Folgenden etwas nachlässig mit M, C, B und A bezeichnet⁴ und als Elemente der Mengen $\mathcal{M} = \mathcal{C} = \mathcal{B} = \{0, 1\}^{512}$ und $\mathcal{A} = \{0, 1\}^{384}$ aufgefasst. \mathcal{P} ist demnach eine Abbildung

$$\mathcal{P} : \mathcal{M} \times \mathcal{C} \times \mathcal{A} \times \mathcal{B} \longrightarrow \mathcal{A} \times \mathcal{B}$$

Für feste Werte $(M, C) \in \mathcal{M} \times \mathcal{C}$ sind die induzierten Abbildungen

$$\begin{aligned} \mathcal{P}_{(M,C)} : \mathcal{A} \times \mathcal{B} &\longrightarrow \mathcal{A} \times \mathcal{B} \\ (A, B) &\longmapsto \mathcal{P}(M, C, A, B) \end{aligned}$$

Permutationen. Die konkrete Definition der Abbildung \mathcal{P} basiert auf der Verwendung von *nicht linear rückgekoppelten Schieberegistern*, kurz *NLFSRs*. Jede Variable wird als NLFSR aufgefasst. Dazu werden die Variablen in Worte (à 32 bit) aufgeteilt, die als *taps* der Register dienen. Die einzelnen taps werden entsprechend mit A_0, \dots, A_{11} bzw. B_0, \dots, B_{15} etc. bezeichnet. Die Besonderheit ist, dass die Updatefunktionen von den anderen Registern abhängen. Abbildung 5 zeigt schematisch die Abhängigkeiten der einzelnen Register. Im ersten Schritt wird A aktualisiert, dabei wird der aktuelle Zustand von B, C und M berücksichtigt. Im zweiten Schritt wird dann B unter Verwendung der Zustände von A, C und M aktualisiert. Die Schlüssel M und C werden unabhängig von den restlichen Registern verändert.

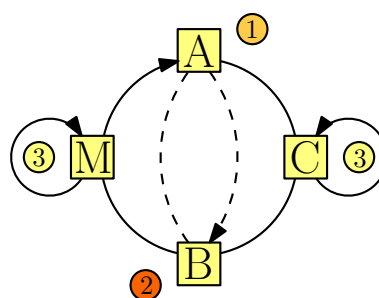


Abbildung 5: Abhängigkeiten der Updatefunktionen, die Zahlen zeigen die Reihenfolge der Ausführung.

⁴ M steht abkürzend für M_ω , entsprechend ist M_i als M_ω^i zu lesen.

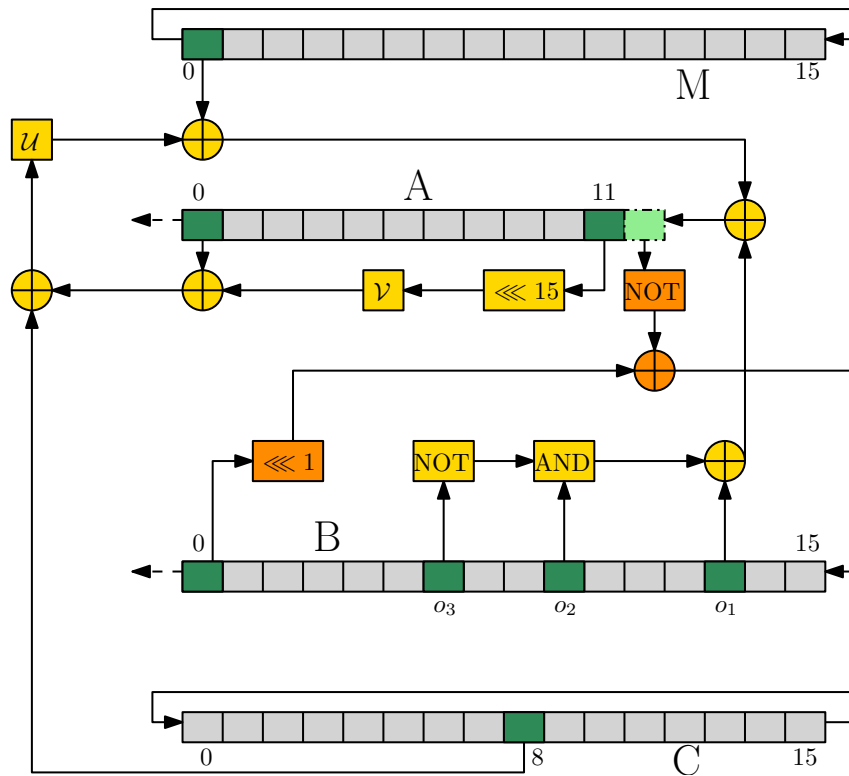


Abbildung 6: Eine Hauptrunde als *verschränkte* Schieberegister, zuerst wird A aktualisiert, dann B und anschließend C und M . Die Updatfunktionen erfüllen die Abhängigkeiten aus Abbildung 5

Diese Prozedur wird über mehrere Runden wiederholt. Die Berechnung von \mathcal{P} lässt sich in drei Phasen unterteilen: Initiale Runden, Hauptrunden, finale Transformation. Es wird sich zeigen, dass jede der drei Phasen invertierbar ist, so dass \mathcal{P} tatsächlich eine Permutation induziert.

2.3.1 Initiale Runden

Die initialen 16 Runden sind sehr einfach und betreffen nur Register B . Jedes der 16 Worte von B wird um 17 Bit zyklisch links rotiert.

Listing 1: Initiale Runden

```
forall 0 ≤ i ≤ 15 do:
    Bi = Bi <<< 17
```

2.3.2 Hauptrunden

Die Hauptrunde wird 32 mal ausgeführt und ist auf den ersten Blick sehr unübersichtlich, da hier die erwähnten gegenseitigen Abhängigkeiten der Updatfunktionen realisiert werden. Abbildung 6 zeigt eine Hauptrunde. \mathcal{U} und \mathcal{V} stehen abkürzend für die (nichtlinearen) Funktionen

$$\begin{aligned} \mathcal{U}(x) &= 3x \pmod{2^{32}} \\ \mathcal{V}(x) &= 5x \pmod{2^{32}} \end{aligned}$$

Die Parameter o_1, o_2 und o_3 sind wie in Abbildung 6 dargestellt auf die Werte

$$(o_1, o_2, o_3) = (13, 9, 6)$$

festgelegt. Mit der Notation $A_{12} = \dots$ für die Updatfunktionen ist gemeint, dass alle taps um eine Position nach links verschoben werden und A_{11} auf den mit A_{12} bezeichneten Wert gesetzt wird. Dabei verfällt der Wert von A_0 . Für die anderen Register gilt entsprechendes, wobei Register C entgegengesetzt rotiert,

dass heißt alle taps nach rechts verschoben werden. Der Wert der Updatefunktion wird entsprechend in C_0 gespeichert. Die Hauptrunden sind dann wie folgt gegeben

32 Runden:

$$\begin{aligned} A_{12} &= 3 \cdot (A_0 \oplus 5 \cdot (A_{11} \lll 15) \oplus C_8) \oplus B_{13} \oplus (B_9 \wedge \overline{B_6}) \oplus M_0 \\ B_{16} &= (B_0 \lll 1) \oplus \overline{A_{11}} \\ M_{16} &= M_0 \\ C_{-1} &= C_{15}. \end{aligned}$$

Das Design der Hauptrunden ist wohl überlegt und in [3] diskutiert, auszugsweise seien einige leicht nachvollziehbare Punkte zitiert. Die Funktionen \mathcal{U} und \mathcal{V} sind als Quelle für Nichtlinearität aufgenommen worden. Dabei wurden aus Geschwindigkeitsgründen sehr einfache Funktionen benutzt. Aus dem selben Grund nutzten die Updatefunktionen auch nur sehr wenig taps der Register. Die Auswahl von (o_1, o_2, o_3) ist so getroffen, dass möglichst schlecht Differentialspuren für \mathcal{P} konstruiert werden können. Die Verwendung von A_0 bzw. B_0 ist essentiell um abzusichern, dass $\mathcal{P}_{(M,C)}$ eine Permutation ist.

2.3.3 Finale Transformation

Nach den Hauptrunden wird noch eine relativ einfache Operation durchgeführt, die hauptsächlich zur Stärkung der Inversen Permutation $\mathcal{P}_{(M,C)}^{-1}$ dient. In Register A wird wortweise der Inhalt von C addiert. Genauer

forall $0 \leq i \leq 35$ **do**:
 $A_j = A_j \boxplus C_{3+j}$

Die Indizes sind dabei modulo 12 bzw. modulo 16 zu verstehen, so dass keine Fehlzugriffe stattfinden. Dies ist offenbar durch wortweise Subtraktion invertierbar und auch leicht als NLFSR realisierbar.

2.3.4 Zusammenfassung

Listing 2 zeigt noch einmal die vollständige Berechnung von $\mathcal{P}_{(M,C)}(A, B)$. Alternativ ist der Quelltext für die Hauptrunden hier nicht für Schieberegister interpretiert.

Listing 2: Vollständige Berechnung von $\mathcal{P}_{(M,C)}(A, B)$

```
# Initiale Runden
forall  $0 \leq i \leq 15$  do:
     $B_i = B_i \lll 17$ 

# Hauptrunden
forall  $0 \leq i \leq 32$  do:
    forall  $0 \leq j \leq 15$  do:
         $A_{j+16i} = 3 \cdot (A_{j+16i} \oplus 5 \cdot (A_{j-1+16i} \lll 15) \oplus C_{8-j}) \oplus B_{j+13} \oplus (B_{j+9} \wedge \overline{B_{j+6}}) \oplus M_j$ 
         $B_j = (B_j \lll 1) \oplus \overline{A_{j+16i}}$ 

# Finale Transformation
forall  $0 \leq i \leq 35$  do:
     $A_j = A_j \boxplus C_{3+j}$ 
```

3 Sicherheitsaspekte

Dieser dritte Teil der Arbeit enthält zusammenfassend Bemerkungen zur erwarteten Sicherheit von Shabal. Die SHA-3 Wettbewerbsausschreibung sieht die Formulierung sogenannter *security claims* - Aussagen zur erwarteten Sicherheit - vor. Die Ausführungen hier sind keineswegs vollständig und sind als Überblick bzw. Ausblick auf die Sicherheit von Shabal und die zur Begründung verwendeten Methoden zu verstehen.

3.1 Grundlegende Sicherheitsbegriffe

Zur Einführung werden in diesem Abschnitt für Sicherheitsbetrachtungen fundamentale Begriffe erklärt. Das sind die Sicherheitsziele wie starke und schwache Kollisionsresistenz, Urbildresistenz und Resistenz gegen length extension attacks. Und außerdem die Begriffe ideale und allgemeine Kryptosysteme.

3.1.1 Sicherheitsziele

Es gibt grundsätzlich drei fundamentale Anforderungen an eine (kryptographische) Hashfunktion, erstens

starke Kollisionsresistenz. Auch einfach *Kollisionsresistenz* bzw. *collision resistance* genannt. Es soll schwer sein zwei verschiedene Nachrichten M und M' zu finden, die den gleichen Hashwert $\mathcal{H}(M) = \mathcal{H}(M')$ haben. Zweitens: ist eine der Nachrichten und damit auch ihr Hashwert bereits vorgegeben, führt dies auf die einfachere Anforderung der

schwachen Kollisionsresistenz. Es soll schwer sein zu einer gegebenen Nachricht M und gegebenen Hashwert $h = \mathcal{H}(M)$ eine weitere Nachricht $M' \neq M$ zu finden, welche den gleichen Hashwert $\mathcal{H}(M') = h$ besitzt. Man spricht daher auch von *zweites Urbild-Resistenz*. Ist schließlich, drittens, ausschließlich der Hashwert h gegeben, heißt diese Anforderung schlicht

Urbildresistenz. Im Englischen auch *preimage resistance* oder *one wayness of \mathcal{H}* . Es soll schwer sein zu einem gegebenen Hashwert $h = \mathcal{H}(M)$ eine Nachricht M' zu finden, die ebenfalls $\mathcal{H}(M') = h$ erfüllt.

Für Merkle-Damgård-Konstruktionen wie Shabal ist noch eine weitere Anforderung wichtig: **Resistenz gegen length extension attacks.** Sind zwei Nachrichten M_1 und M_2 (gleicher Länge $|M_1| = |M_2|$) mit $\mathcal{H}(M_1) = \mathcal{H}(M_2)$ gegeben, so gilt für jede weitere Nachricht M

$$\mathcal{H}(M_1 \| M) = \mathcal{H}(M_2 \| M).$$

$M_1 \| M$ bezeichnet hier die Konkatenation der Nachrichten. Das heißt, man kann aus einer Kollision leicht viele weitere Kollisionen erzeugen. Mit vielen zu Verfügung stehenden Kollisionen ist es wesentlich einfacher, die drei Anforderungen von oben zu widerlegen.

3.1.2 Ideale Kryptosysteme

Für theoretische Betrachtungen ist es oft hilfreich idealisierte Kryptosysteme zu betrachten, welche die geforderten Anforderungen per Definition erfüllen. Klassische Varianten sind das sogenannte Zufallsorakel und die ideale Chiffre. Sie sind die idealen Varianten von kryptographischen Hashfunktionen bzw. symmetrischen Verschlüsselungen. Die folgenden Erklärungen sind keine ausgereiften Definitionen sondern dienen nur zur Verdeutlichung der Ideen.

Ein **Zufallsorakel** ist eine Hashfunktion. Aus einer Anfrage in Form einer Nachricht wird ein rein zufällig gewählter Hashwert generiert. Die Antworten sind dabei in entsprechendem Wertebereich gleichverteilt und unabhängig voneinander. Die Antworten des Zufallsorakels sind natürlich konsistent, zwei Anfragen mit der gleichen Nachricht ergeben stets den selben Hashwert. Es ist klar, dass ein Zufallsorakel den obigen Sicherheitszielen genügt. Kollisionen sind nur durch Testen vieler Nachrichten zu finden.

Eine **Ideale Chiffre** ist eine Blockchiffre der Form $\mathcal{B} \in \text{Sym}(\mathcal{M})^{\mathcal{K}}$. Hier bezeichnet \mathcal{M} den Nachrichten- und Kryptotextraum und \mathcal{K} den Schlüsselraum. Für gegebenen Schlüssel $k \in \mathcal{K}$ erhält man eine Bijektion $\mathcal{B}_k : \mathcal{M} \rightarrow \mathcal{M}$, die gegebenen Nachrichten ihren Kryptotext zuordnet. Die Bijektion \mathcal{B}_k wird in Abhängigkeit von k zufällig unter Gleichverteilung gewählt. Es ist klar, dass die Kenntnis von ausschließlich \mathcal{B}_k keine Rückschlüsse auf k zulässt.

3.1.3 Kryptosysteme

Neben den idealen Kryptosystemen ist es nützlich ein Rahmenwerk für allgemeine Kryptosysteme zu haben. Es soll spezifiziert werden, wie sich ein kryptographischer Algorithmus verhält und wie man ihn benutzt, ohne jedoch auf seine Realisierung oder konkrete Implementierung Bezug zu nehmen. Die Definitionen in diesem Abschnitt entstammen [2].

Ein **Kryptosystem** \mathcal{C} lässt sich als Folge von bedingten Wahrscheinlichkeitsverteilungen $(P_n)_{n \in \mathbb{N}}$ auffassen. $P_i = P_{Y_i | X^i Y^{i-1}}$ bezeichnet die Verteilung der i -ten Ausgabe Y_i unter den Bedingungen, dass $X^i = (X_1, \dots, X_i)$ und $Y^{i-1} = (Y_1, \dots, Y_{i-1})$ die i Eingaben und ersten $i - 1$ Ausgaben bekannt sind. Die X_i und Y_i sind hierbei als Zufallsvariablen zu verstehen.

Beispielsweise würde für das oben beschriebene Zufallsorakel P_1 eine Gleichverteilung auf dem Raum der zugelassenen Hashwerte sein. Tatsächlich wird die erste Ausgabe y_1 unabhängig von der übermittelten Nachricht x_1 gewählt. Für Y_2 ergibt sich $Y_2 = y_1$, falls $X_2 = x_1$ und $X_1 = x_1$ bzw. im Fall $x_1 \neq x_2$ und $X_2 = x_2$ und $X_1 = x_1$ ist Y_2 gleichverteilt. Entsprechend können die weiteren P_i beschrieben werden.

In der Regel hängen Kryptosysteme von einem Sicherheitsparameter $k \in \mathcal{N}$ ab, beispielsweise von der Schlüssellänge oder der Rundenzahl in entsprechenden Algorithmen. Um dies abzubilden betrachtet man ein System $(\mathcal{C}_k)_{k \in \mathbb{N}}$ von Kryptosystemen in obigem Sinne.

Um Angriffsszenarien zu studieren betrachtet man Kryptosysteme mit Schnittstellen. Es wird in öffentliche und private Schnittstellen unterschieden. Die *privaten* Schnittstellen eines Kryptosystems \mathcal{C} modellieren den vorgesehenen Umgang mit dem System, sie stehen nur den ehrlichen Benutzern zur Verfügung und werden mit \mathcal{C}^{priv} bezeichnet. Die *öffentlichen* Schnittstellen \mathcal{C}^{pub} stehen jeder Partei frei zu Verfügung. Insbesondere interagieren Angreifer mit dem System über die öffentliche Schnittstelle, vgl. Abbildung 8. Mehrere Systeme kann man zu neuen Systemen zusammensetzen, indem man die entsprechenden Schnittstellen miteinander verbindet. Die Trennung in öffentliche und private Schnittstellen dient dazu, Wissensvorteile des Angreifers zu modellieren. Das ist ein entscheidender Punkt im Vergleich der Sicherheit kryptographischer Systeme. Im Beispiel der idealen Systeme von oben stimmen öffentliche und private Schnittstellen überein, da es kein Zusatzwissen für den Angreifer geben kann.

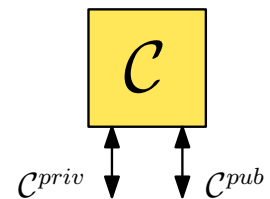


Abbildung 7: Kryptosystem \mathcal{C} mit öffentlicher Schnittstelle \mathcal{C}^{pub} und privater Schnittstelle \mathcal{C}^{priv}

3.2 Vergleich kryptographischer Systeme

Die Sicherheit eines kryptographischen Verfahrens an sich zu beurteilen ist schwierig. Der entscheidende Schritt ist, die Sicherheit zweier Systeme zu vergleichen. Auf diese Weise kann man ein gegebenes Kryptosystem zu einem idealen in Bezug setzen und so seine Sicherheit beurteilen.

3.2.1 Ununterscheidbarkeit von Kryptosystemen

In der Terminologie der folgenden Abschnitte bezeichnen \mathcal{E} oder \mathcal{D} Kryptosysteme mit den Ausgabewerten 0 oder 1. Sie werden als Umgebung (*Environment*) oder Unterscheider (*Distinguisher*) bezeichnet. Angreifer und sonstige Parteien sind Algorithmen, also Turingmaschinen, aufgefasst als Systeme, die über die entsprechenden Schnittstellen mit anderen Systemen interagieren.

Seien $\mathcal{C} = (\mathcal{C}_k)_{k \in \mathbb{N}}$ und $\mathcal{C}' = (\mathcal{C}'_k)_{k \in \mathbb{N}}$ zwei Kryptosysteme *ohne* öffentliche Schnittstellen. \mathcal{C} und \mathcal{C}' heißen **ununterscheidbar**, falls für jeden Unterscheider \mathcal{D} der Vorteil

$$|\text{Prob}[\mathcal{D}(\mathcal{C}_k) = 1] - \text{Prob}[\mathcal{D}(\mathcal{C}'_k) = 1]|$$

vernachlässigbar in k ist.

Die Idee der Definition ist, dass sich die beiden Systeme von außen betrachtet nicht unterscheiden lassen. Stellt man sich eine Blackbox mit \mathcal{C} oder \mathcal{C}' im Inneren vor, so kann man aus regulären Anfragen und ohne Zusatzwissen nicht entscheiden, mit welchem System im Inneren die Blackbox arbeitet. Praktisch ist diese Definition nicht relevant, da fast immer Zusatzwissen, also öffentliche Schnittstellen bereitstehen. Zum Beispiel ist bei Hashalgorithmen der Algorithmus bekannt. Mit dieser Information ist es sehr leicht, ihn anhand seiner Antworten von anderen zu unterscheiden. Der Begriff der Ununterscheidbarkeit wird unten wesentlich zum Begriff der Undifferenzierbarkeit verallgemeinert.

Folgende Definition ist grundlegend. Seien nun auch öffentliche Schnittstellen für die Systeme \mathcal{C} und \mathcal{C}' erlaubt. \mathcal{C} heißt (**komplexitätstheoretisch**) **mindestens so sicher** wie \mathcal{C}' , falls es für jede Umgebung \mathcal{E} und für jeden Angreifer \mathcal{A} für \mathcal{C} einen Angreifer \mathcal{A}' für \mathcal{C}' gibt, sodass der Unterschied in den Verteilungen

$$|\text{Prob}[\mathcal{E}(\mathcal{C}_k^{priv}, \mathcal{A}(\mathcal{C}_k^{pub})) = 1] - \text{Prob}[\mathcal{E}(\mathcal{C}'_k^{priv}, \mathcal{A}'(\mathcal{C}'_k^{pub})) = 1]|$$

vernachlässigbar in k ist (wobei $\mathcal{E}, \mathcal{A}, \mathcal{A}'$ effiziente Algorithmen sind).

Die Bezeichnung ist $\mathcal{C} \succ \mathcal{C}'$ bzw. $\mathcal{C} \succ_c \mathcal{C}'$ im komplexitätstheoretischen Fall. Die Bezeichnungen in der Definition bedeuten, dass die Systeme wie in Abbildung 8 miteinander verbunden sind. Die Definition sagt, dass alles was ein Angreifer \mathcal{A} bei \mathcal{C} erreichen kann, auch ein entsprechender Angreifer \mathcal{A}' bei \mathcal{C}' erreichen kann. Mit anderen Worten, die Kryptosysteme \mathcal{C} und \mathcal{C}' unterscheiden sich in Interaktion mit einem Angreifer nur so gering, dass dieser Unterschied durch Modifikation des Angreifers ausgeglichen werden kann. Letzteres in dem Sinne, dass sich die Szenarien in jeder Umgebung gleich darstellen. Die Systeme $(\mathcal{C}^{priv}, \mathcal{A}(\mathcal{C}^{pub}))$ und $(\mathcal{C}'^{priv}, \mathcal{A}'(\mathcal{C}'^{pub}))$ sind in obigem Sinne ununterscheidbar.

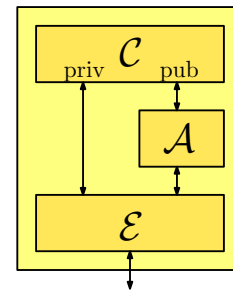


Abbildung 8: Kryptosystem \mathcal{C} mit Angreifer \mathcal{A} und Umgebung \mathcal{E}

Ein Kryptosystem \mathcal{C} ist nun **genauso sicher** wie \mathcal{C}' , falls sowohl $\mathcal{C} \succ \mathcal{C}'$ als auch $\mathcal{C} \prec \mathcal{C}'$ gilt. Entsprechendes gilt für komplexitätstheoretische Sicherheit.

Ein wichtiges **Theorem** sagt:

Zwei Kryptosysteme \mathcal{S} und \mathcal{T} sind ununterscheidbar genau dann, wenn für jedes weitere System $\mathcal{C}(\cdot)$ das System $\mathcal{C}(\mathcal{T})$ genauso sicher ist wie $\mathcal{C}(\mathcal{S})$.

Dies legt folgende Strategie zum Beweis der Sicherheit eines Kryptosystems nahe, vgl [1]. Im Fall von Hashfunktionen sind zwei Dinge zu beweisen. Erstens: das reale System ist ununterscheidbar von einem idealisierten System, welches zum Beispiel idealisierte Komponenten benutzt. Und zweitens: das idealisierte System ist ununterscheidbar von einem idealen (perfekten) System. Im Beispiel von Shabal

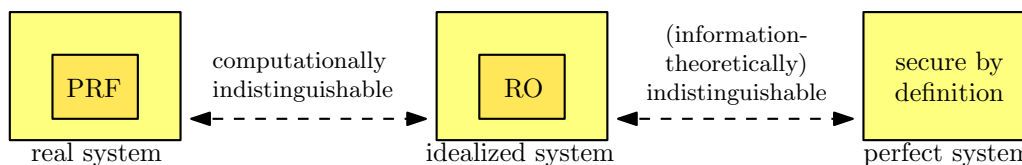


Abbildung 9: Beweisschema für Sicherheit einer Hashfunktion

wäre im *idealisierten Shabal* die schlüsselabhängige Permutation durch eine ideale Blockchiffre ersetzt. Das perfekte System wäre ein Zufallsorakel. Es wäre also zu zeigen, dass man Shabal ($\mathcal{SH}(\mathcal{P})$) nicht von der idealisierten Variante $\mathcal{SH}(\mathcal{B})$ die statt der Permutation \mathcal{P} eine ideale Blockchiffre \mathcal{B} benutzt unterscheiden kann. Und es wäre zu zeigen, dass man auch $\mathcal{SH}(\mathcal{B})$ nicht von einem Zufallsorakel \mathcal{RO} unterscheiden kann. Es ist nicht zu erwarten, dass einer der Beweise ohne starke Einschränkungen zu führen ist. Außerdem gibt es öffentliche Schnittstellen, so dass das Theorem von oben nicht anwendbar ist. Die Begriffe müssen zunächst verfeinert und eine analoge Variante des Theorems gefunden werden.

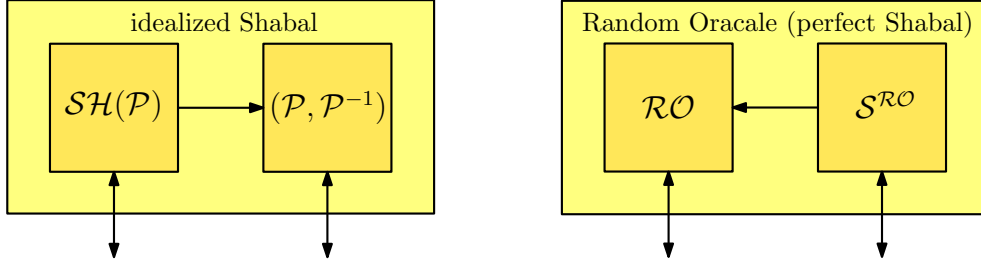


Abbildung 10: Idealisierter und perfekter Shabal, als Kryptosysteme mit öffentlichen und privaten Schnittstellen

3.2.2 Undifferenzierbarkeit

Seien $\mathcal{C} = (\mathcal{C}_k)_{k \in \mathbb{N}}$ und $\mathcal{C}' = (\mathcal{C}'_k)_{k \in \mathbb{N}}$ zwei Kryptosysteme mit öffentlichen Schnittstellen. \mathcal{C} heißt **undifferenzierbar** von \mathcal{C}' (\mathcal{C} is indifferntiable from \mathcal{C}'), falls für jeden Entscheider \mathcal{D} ein System \mathcal{P} existiert, so dass der Vorteil für \mathcal{D}

$$|\text{Prob}[\mathcal{D}(\mathcal{C}_k^{priv}, \mathcal{C}_k^{pub}) = 1] - |\text{Prob}[\mathcal{D}(\mathcal{C}'_k^{priv}, \mathcal{P}(\mathcal{C}'_k^{pub})) = 1]|$$

vernachlässigbar in k ist.

Die Bezeichnung ist $\mathcal{C} \sqsubset \mathcal{C}'$. Sind \mathcal{D} und \mathcal{P} effiziente Algorithmen, so spricht man von komplexitätstheoretischer Undifferenzierbarkeit und schreibt $\mathcal{C} \sqsubset_c \mathcal{C}'$.

Der Begriff der Undifferenzierbarkeit ist eine direkte Verallgemeinerung von Ununterscheidbarkeit. Falls \mathcal{S} und \mathcal{T} keine öffentliche Schnittstellen besitzen, so fallen Ununterscheidbarkeit und symmetrische Undifferenzierbarkeit zusammen. Wie in [2] vorgeschlagen, könnte ein erster Ansatz zur Definition auf das Zwischensystem \mathcal{P} verzichten. Es stellt sich jedoch heraus, dass dies auf einen zu strengen Begriff führt, so dass das Theorem von oben sich nicht adäquat verallgemeinern lässt. Es gäbe in diesem Fall Kryptosysteme, die zwar gleich sicher wären, aber nicht undifferenzierbar. Mit der angegebenen Definition erhält man jedoch

Theorem: Seien \mathcal{S} und \mathcal{T} zwei Kryptosysteme und sei \mathcal{S} undifferenzierbar von \mathcal{T} , so ist $\mathcal{C}(\mathcal{S})$ mindestens so sicher wie $\mathcal{C}'(\mathcal{T})$ für jedes weitere System $\mathcal{C}(\cdot)$, also

$$\mathcal{S} \sqsubset \mathcal{T} \iff \forall \mathcal{C}(\cdot) : \mathcal{C}(\mathcal{S}) \succ \mathcal{C}(\mathcal{T}).$$

Mit diesem Theorem kann man nun die in Abbildung 9 dargestellte Strategie wieder aufgreifen.

3.3 Anwendungen auf Shabal

Um die entwickelte Strategie auf Shabal anzuwenden, muss Shabal als Kryptosystem mit seinen öffentlichen und privaten Schnittstellen spezifiziert werden. Die private Schnittstelle bietet naheliegend die Berechnung von Hashwerten zu gegebenen Nachrichten an. Die öffentliche Schnittstelle erlaubt neben dem hashen von Nachrichten auch noch Auswertungen der Permutationen \mathcal{P} und \mathcal{P}^{-1} zu gegebenen Parametern. In diesem Sinne ist dann der mode of operation von Shabal als Kryptosystem $\mathcal{SH}(\cdot)$ zu verstehen. Zusammen mit der schlüsselabhängigen Permutation \mathcal{P} , ebenfalls aufgefasst als Kryptosystem, ergibt sich der gesamte Algorithmus $\mathcal{SH}(\mathcal{P})$. In einer idealisierten Variante wird \mathcal{P} durch eine ideale Blockchiffre ersetzt. In der perfekten Variante wird nun Shabal durch ein Zufallsorakel ersetzt. Um jedoch die gleichen Schnittstellen anzubieten, ist nun ein System $\mathcal{S}^{\mathcal{RO}}$ nötig, welches unter Verwendung des Orakels eine Ideale Blockchiffre simuliert. Abbildung 10 fasst die Situation zusammen.

3.3.1 Undifferenzierbarkeit von einem Zufallsorakel

Im Wettbewerbsbeitrag [3] wird nun unter Anderem folgender Satz bewiesen.

Satz: Es existiert eine Simulation \mathcal{S} von $(\mathcal{P}, \mathcal{P}^{-1})$, sodass jeder Unterscheider \mathcal{D} nach N Fragen an die öffentliche Schnittstelle einen Vorteil,

$$|\Pr[\mathcal{D}((\mathcal{P}, \mathcal{P}^{-1})) = 1] - \Pr[\mathcal{D}(\mathcal{S}^{\mathcal{RO}}) = 1]| \leq N(4N - 3)2^{-896}$$

hat. \mathcal{S} stellt dabei maximal N Fragen an \mathcal{RO} und läuft in Zeit $O(N^2)$.

Daraus leiten die Autoren ab, dass zumindest ein idealisierter Shabal mit höchstens 2^{488} Auswertungen von \mathcal{P} oder \mathcal{P}^{-1} von einem Zufallsorakel undifferenzierbar ist.

3.3.2 Security Claims

Die Ausschreibung für SHA-3 verlangt die Angaben sogenannter Security Claims und Angaben zu ihrer Plausibilität. Im Fall von Shabal werden mit Aussagen wie aus dem vorhergehendem Abschnitt folgende Behauptungen begründet. Normiert man den Aufwand der Auswertung einer Nachrichtenrunde auf 1 so gilt

- Das Finden einer Kollision hat mindestens den Aufwand $\sqrt{2^{l_h}}$
- Das finden eines Urbilds zu gegebenem Hashwert hat mindestens den Aufwand 2^{l_h}
- Das Finden eines zweiten Urbilds unter Nachrichten der Länge $\leq 2^k$ hat mindestens den Aufwand $2^{l_h - k}$
- Jede length extension attack hat mindestens den Aufwand 2^{256}

l_h bezeichnet hierbei wieder die Länge des Hashwertes. $l_h \in \{192, 224, 256, 384, 512\}$.

Literatur

- [1] Ueli Maurer. Indistinguishability of random systems. In *In Advances in Cryptology — EUROCRYPT '02, volume 2332 of LNCS*, pages 110–132. Springer-Verlag, 2002.
- [2] Ueli Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *Theory of Cryptography - TCC 2004, Lecture Notes in Computer Science*, pages 21–39. Springer-Verlag, 2003.
- [3] Saphir project, www.shabal.com. *Shabal, a Submission to NIST's Cryptographic Hash Algorithm Competition*, 1.0 edition, 01 2009.