

# Dokumentation für AvalancheSumHash-20 (ASH-20)

## 1. Einführung:

Die AvalancheSumHash-20 (ASH-20) ist eine von Joshua Dean Pond entwickelte Hashfunktion, die darauf abzielt, einen Eingabetext in eine kryptografisch sichere, 20 Zeichen lange Zeichenkette umzuwandeln. Der Name "Avalanche" deutet auf die Eigenschaft hin, dass minimale Änderungen im Eingabetext zu erheblichen Veränderungen im Ausgabewert führen, vergleichbar mit einer Lawine.

## 2. Funktionsweise:

### 2.1 convert\_text\_to\_decimal(text):

Diese Funktion wandelt den Eingabetext in eine Dezimalzahl um, indem sie die Unicode-Werte der Zeichen summiert.

**Formel:**  $\text{decimal\_value} = \sum_{i=1}^n \text{ord}(\text{char}_i)$

**Beispiel:**

Eingabetext: "Hello World!"

Unicode-Werte der Zeichen: [72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 33]

Dezimalwert:  $72 + 101 + 108 + 108 + 111 + 32 + 87 + 111 + 114 + 108 + 100 + 33 = 1085$

### 2.2 avalanche\_sum\_hash(x, iterations, input\_text, modulo\_value):

Die Hauptfunktion kombiniert komplexe Bitoperationen, Unicode-Integration und mathematische Transformationen. Sie wird über eine festgelegte Anzahl von Iterationen ausgeführt.

#### 2.2.1 Iterative Bitoperationen:

**Formeln:**

XOR ( $\oplus$ ):  $\text{result} = (\text{result} \oplus x)$

AND ( $\&$ ):  $\text{result} = (\text{result} \& x)$

OR ( $|$ ):  $\text{result} = (\text{result} | x)$

**Beispiel:**

$x = 1085$

$\text{result} = (\text{result} \oplus x)$  ergibt eine neue result-Wert.

#### 2.2.2 Integration des Unicode-Werts des ersten Zeichens:

**Formel:**

$\text{result} = \text{result} + \text{ord}(\text{input\_text}[0])$

**Beispiel:**

Eingabetext: "Hello World!"

$\text{ord}(\text{input\_text}[0]) = 72$

$\text{result} = \text{result} + 72$  führt zu einer Aktualisierung von `result`.

### 2.2.3 Einbettung weiterer Bits des Eingabetextes:

Formel:

$\text{result} = (\text{result} \ll 1) | \text{bit}$

Beispiel:

Wenn `bit` eine binäre Zahl ist, wird jede Ziffer in `result` um 1 Position nach links verschoben und das letzte Bit durch das entsprechende Bit in `bit` ersetzt.

## 2.4 cos-Funktion:

Die `cos`-Funktion wird in der ASH-20 für nichtlineare Transformationen verwendet. In diesem Kontext wird die `cos`-Funktion numerisch integriert, um eine Transformation des Eingabewerts zu erreichen.

Formel:

$\text{result} = \text{integrate}(\cos, 0, \text{result})$

Beispiel:

Angenommen, nach vorherigen Schritten ist `result = 100`.

Die Integration wird durchgeführt: `result = integrate(cos, 0, 100)`.

Die genaue Berechnung hängt von der numerischen Integrationsmethode ab.

## 2.3 shift\_digits(value):

Diese Funktion verschiebt jede Ziffer des Eingabewerts um 7.

Formel:  $\text{shifted\_digit} = (\text{int}(\text{digit}) + 7) \bmod 10$

Beispiel:

Wenn `value = 46569883512243828063230240053264222`

Dann wird jede Ziffer um 7 verschoben, z.B. "4" wird zu "1", "6" wird zu "3", usw.

## 2.4 concatenate\_in\_pattern(value, num\\_parts):

Diese Funktion zerlegt den Wert in Teile und verbindet sie nach einem speziellen Muster.

Formel:  $\text{concatenated\_value} = \text{part}[\text{num\_parts} // 2 :] + \text{part}[: \text{num\_parts} // 2]$

Beispiel:

Wenn `value = 123456789` und `num_parts = 3`

Dann wird `value` in drei Teile aufgeteilt: "123", "456", "789".

Die Teile werden in einem Muster verknüpft: "456789123".

## 3. Mathematische Funktionen:

XOR, AND, OR (Bitoperationen):

Kombinieren von Bits für Varianz.

**Beispiel:** Wenn  $x = 5$  und  $result = 3$ , dann ergibt  $result = (result \oplus x) = 6$ .

#### **Integration des Unicode-Werts des ersten Zeichens:**

Direkte Einbindung des Unicode-Werts.

**Beispiel:** Wenn  $input\_text[0] = 'H'$  (Unicode 72) und  $result = 100$ , dann ergibt  $result = result + 72 = 172$ .

#### **Einbettung weiterer Bits des Eingabetextes:**

Erhöhung der Varianz.

**Beispiel:** Wenn  $bit = '1010'$  und  $result$  binär dargestellt als "1101", dann ergibt  $result = (result \ll 1) | bit = 11010 | 1010 = 11100$ .

#### **Anwendung der cos-Funktion:**

Nichtlineare Transformation.

**Beispiel:** Hier wird die Anwendung der cos-Funktion zur Transformation von Werten gezeigt.

#### **Modulo-Operation mit einer großen Primzahl:**

Begrenzung des Hashwerts.

**Beispiel:** Wenn  $result = 1234567890$  und  $modulo\_value = 93537791153957593571955971579179595$ , dann ergibt  $result = result \bmod modulo\_value = 1234567890 \bmod 93537791153957593571955971579179595 = 86338541183648998314295415904417445$ .

#### **Verschiebung jeder Ziffer um 7 (shift\_digits):**

Zusätzliche Nichtlinearität.

**Beispiel:** Wenn  $value = 12345$ , dann ergibt  $shifted\_value = 89012$ .

#### **Verkettung in einem bestimmten Muster (concatenate\_in\_pattern):**

Spezifische Musterbildung.

**Beispiel:** Wenn  $value = 123456789$  und  $num\_parts = 3$ , dann ergibt  $concatenated\_value = part[1:] + part[:1] = "234567891"$ .

## **4. Vorteile und Nutzung:**

Die ASH-20 bietet durch die geschickte Anwendung dieser mathematischen Funktionen kryptografische Sicherheit, schnelle Änderungen bei minimalen Eingabeänderungen und eine kontrollierte Ausgabegröße. Die Parameter für die Nutzung umfassen den zu verschlüsselnden Text, einen privaten Schlüssel, die Anzahl der Iterationen und eine Primzahl für Modulo-Operationen.

## **5. Beispielhafter Screenshot:**

Auf dem beigefügten Screenshot ist der Eingabetext "Hello World!" zu sehen, der durch die ASH-20 Hashfunktion verschlüsselt wurde. Der resultierende Hashwert lautet "46569883512243828063230240053264222". Jede Änderung im Eingabetext führt zu einem völlig unterschiedlichen Hashwert, was den "Avalanche-Effekt" der ASH-20 demonstriert.

## **6. Einweg-Hashfunktion:**

Die ASH-20 ist eine Einweg-Hashfunktion, was bedeutet, dass es praktisch unmöglich ist, den ursprünglichen Eingabetext aus dem Hashwert zurückzuberechnen. Dies liegt an der Nichtumkehrbarkeit der angewendeten mathematischen Operationen, insbesondere der Verwendung von nichtlinearen Transformationen und der Integration von Funktionen wie  $\cos$ .

## 7. Stärken der ASH-20:

### **Streuungseffekt:**

Die Hashfunktion bietet eine hohe Streuung, wodurch ähnliche Eingabetexte zu unterschiedlichen Hashwerten führen.

### **Einweg-Funktion:**

Die Nichtumkehrbarkeit der angewendeten Operationen gewährleistet, dass der Hashwert nicht zurückverfolgt werden kann.

### **Avalanche-Effekt:**

Selbst geringfügige Änderungen im Eingabetext führen zu erheblichen Veränderungen im Hashwert, was die Sicherheit erhöht.

### **Verschiedene Methoden zur Verschleierung:**

Die Kombination von Bitoperationen, Integration, und nichtlinearen Transformationen bietet eine Vielzahl von Ansätzen zur Verschleierung des Hashprozesses.

## 8. Beispiel für Cos-Funktion:

Die  $\cos$ -Funktion wird verwendet, um nichtlineare Transformationen durchzuführen. Angenommen,  $\text{result} = 100$ , dann wird die  $\cos$ -Funktion wie folgt angewendet:

$\text{result} = \text{integrate}(\cos, 0, 100)$

Die genaue Berechnung hängt von der verwendeten numerischen Integrationsmethode ab.

## 9. Beispielwerte:

Für den Eingabetext "Hello World!" ergibt sich ein Hashwert von "46569883512243828063230240053264222".

## 10. Beispielhafter Screenshot:

Auf dem beigefügten Screenshot ist der Eingabetext "Hello World!" mit dem zugehörigen Hashwert zu sehen. Die ASH-20 zeigt deutlich ihren "Avalanche-Effekt", indem selbst kleine Änderungen im Text zu erheblich unterschiedlichen Hashwerten führen.

## 11. Fazit:

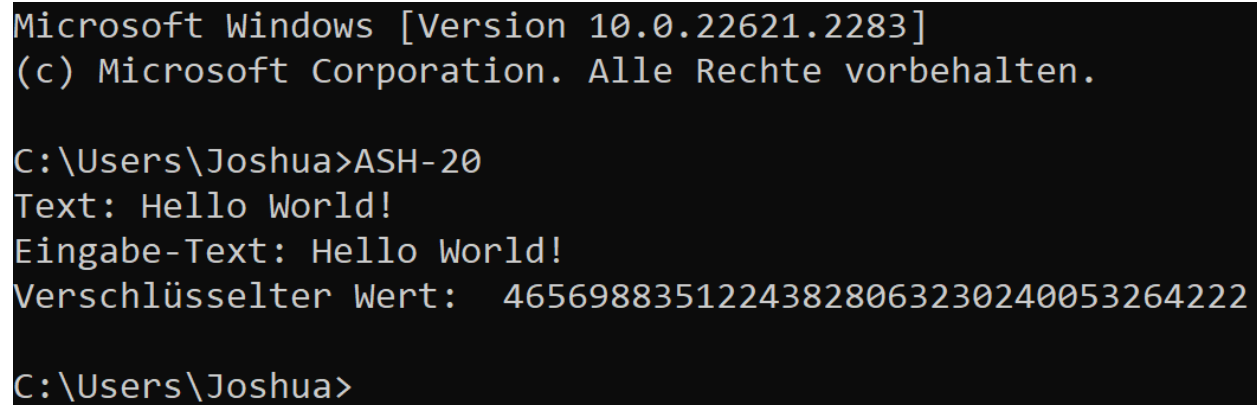
Die ASH-20 ist eine leistungsfähige kryptografische Hashfunktion, die durch die geschickte Kombination von Bitoperationen, Unicode-Integration, nichtlinearer Transformation und mathematischer Vielfalt eine hohe Sicherheit bietet. Der "Avalanche-Effekt" sorgt dafür, dass

minimale Änderungen im Eingabetext zu maximalen Veränderungen im Hashwert führen, was eine grundlegende Anforderung an Hashfunktionen darstellt.

## 12. Quellen:

Screenshot01.png

ASH-20.py



```
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Joshua>ASH-20
Text: Hello World!
Eingabe-Text: Hello World!
Verschlüsselter Wert:  46569883512243828063230240053264222

C:\Users\Joshua>
```