Secure Hash Algorithm SHA-256

Chi Trung Nguyen *T-Systems*



20. Juni 2012



IMPLEMENTIERUNG

ANWENDUNG

00

AUSBLICK

0

INHALT

EINFÜHRUNG

EINFÜHRUNG

Was ist ein Hash?

GESCHICHTE

GESCHICHTE

SHA Allgemein

SHA-0

Verwendungszweck

Sicherheitslücken

AUSBLICK

SHA-3

WAS IST EIN HASH?

▶ deutsch: "zerhacken", "verstreuen"



Secure Hash Algorithm 3 of

WAS IST EIN HASH?

- ► deutsch: "zerhacken", "verstreuen"
- ► Hashfunktion oder Streuwertfunktion erstellt aus beliebiger großer Quellmenge eine immer gleich große Zielmenge

$$f(x) = f(x')$$



Secure Hash Algorithm 4 c

WAS IST EIN HASH?

- ► deutsch: "zerhacken", "verstreuen"
- ► Hashfunktion oder Streuwertfunktion erstellt aus beliebiger großer Quellmenge eine immer gleich große Zielmenge
 - f(x) = f(x')
- ► Einwegfunktion



Secure Hash Algorithm

SHA ALLGEMEIN

► 1993 vom National Institute of Standards(NIST) als ein U.S. Federal Information Processing Standard (FIPS) veröffentlicht



Secure Hash Algorithm 6 o

•0000

SHA ALLGEMEIN

- ▶ 1993 vom National Institute of Standards(NIST) als ein **U.S. Federal Information Processing Standard (FIPS)** veröffentlicht
- ► Gruppe von kryptologischer Hashfunktionen
 - ► SHA-0
 - ► SHA-1
 - ► SHA-2
 - ► SHA-3



Secure Hash Algorithm

SHA-0

► 1993 veröffentlicht



Secure Hash Algorithm 8 of 38

SHA-0

- ► 1993 veröffentlicht
- ▶ Bestandteil des Digital Signature Algorithms (DSA) für Digital Signature Standard (DSS)



Secure Hash Algorithm 9 of 38

- 1. 2005 von Xiaoyun Wang, Yiqun Lisa Yin und Hongbo Yu an der Shandong University in China gebrochen
- 2. Ihnen war es gelungen, den Aufwand zur Kollisionsberechnung von 2^{80} auf 2^{69} zu verringern
- 3. es wurde ein rechtsshift durch ein linksshift ersetzt
- 4. August 2005, wurde von Xiaoyun Wang, Andrew Yao und Frances Yao auf der Konferenz CRYPTO 2005 ein weiterer, effizienterer Kollisionsangriff auf SHA-1 vorgestellt, welcher den Berechnungsaufwand auf 2⁶³ reduziert

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG ANWENDUNG AUSBLICK
0 00•00 00000000 00 0

SHA-1

► 1995 veröffentlicht



Secure Hash Algorithm 10 o

- 1. 2005 von Xiaoyun Wang, Yiqun Lisa Yin und Hongbo Yu an der Shandong University in China gebrochen
- 2. Ihnen war es gelungen, den Aufwand zur Kollisionsberechnung von 2^{80} auf 2^{69} zu verringern
- 3. es wurde ein rechtsshift durch ein linksshift ersetzt
- 4. August 2005, wurde von Xiaoyun Wang, Andrew Yao und Frances Yao auf der Konferenz CRYPTO 2005 ein weiterer, effizienterer Kollisionsangriff auf SHA-1 vorgestellt, welcher den Berechnungsaufwand auf 2⁶³ reduziert

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG ANWENDUNG AUSBLICK
O 00•00 00000000 00 0

SHA-1

- ▶ 1995 veröffentlicht
- ► aufgrund Designfehler in SHA-0



Secure Hash Algorithm 11 of

IMPLEMENTIERUNG

► 2002 veröffentlicht

GESCHICHTE

00000

ANWENDUNG

00

Einführung

AUSBLICK

0

SHA-2

- ► 2002 veröffentlicht
- existiert in mehreren Bit Variante



Secure Hash Algorithm 13 of 38

- 2⁶⁴ bits = 2 147 483 648 gigabytes
 auch wenn 2¹²⁸ bits an daten mglich sind, werden 2⁶⁴ in der realität nicht überschritten

HRUNG	GESCHICHTE	Implementierung	Anwendung	Ausblick
	00000	00000000	00	0

Tabelle: Secure Hash Algorithmus Eigenschaften

EINFÜE

Algorithmus	Message	Block Größe(bits)	Word Größe(bits)	Message Digest	
	Größe(bits)			Größe(bits)	
SHA-1	$< 2^{64}$	512	32	160	
SHA-224	$< 2^{64}$	512	32	224	
SHA-256	$< 2^{64}$	512	32	256	
SHA-384	$< 2^{128}$	1024	64	384	
SHA-512	$< 2^{128}$	1024	64	512	



Secure Hash Algorithm

FUNKTIONEN

EINFÜHRUNG

$$Ch(E, F, G) = (E \land F) \oplus (\neg E \land G)$$

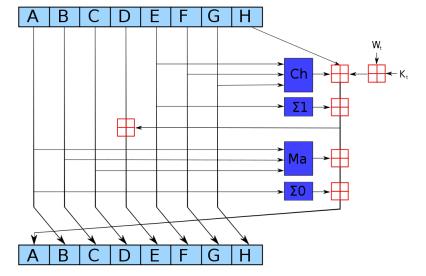
$$Maj(A, B, C) = (A \land B) \oplus (A \land C) \oplus (B \land C)$$

$$\Sigma_0 = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$

$$\Sigma_1 = (A \ggg 6) \oplus (A \ggg 11) \oplus (A \ggg 25)$$

15 of 38

DARSTELLUNG DES ALGORITHMUS



PSEUDOCODE

► Initialize variables (first 32 bits of the fractional parts of the square roots of the first 8 primes 2..19):

$$h[0..7] := 0x6a09e667, [...], 0x5be0cd19$$

PSEUDOCODE

▶ Initialize variables (first 32 bits of the fractional parts of the square roots of the first 8 primes 2..19):

```
h[0..7] := 0x6a09e667, [...], 0x5be0cd19
```

► Initialize table of round constants (first 32 bits of the fractional parts of the cube roots of the first 64 primes 2..311):

$$k[0..63] := 0x428a2f98, [...], 0xc67178f2$$



Secure Hash Algorithm 18 o

▶ bit 1 zur *message* hinzufügen

- ▶ bit 1 zur *message* hinzufügen
- ▶ anzahl von k bits 0 hinzufügen, wobei k die kleinst mögliche Zahl >= 0, so dass die Länge der message (in bits) Modulo 512 minus 64 bits ist

- ▶ bit 1 zur *message* hinzufügen
- ▶ anzahl von k bits 0 hinzufügen, wobei k die kleinst mögliche Zahl >= 0, so dass die Länge der *message* (in bits) Modulo 512 minus 64 bits ist
- ► Länge der *message* (vor dem Preprocessing), in bits, als 64-bit big-endian integer hinzufgen

21 of 38

- ▶ bit 1 zur *message* hinzufügen
- ▶ anzahl von k bits 0 hinzufügen, wobei k die kleinst mögliche Zahl >= 0, so dass die Länge der message (in bits) Modulo 512 minus 64 bits ist
- ► Länge der *message* (vor dem Preprocessing), in bits, als 64-bit big-endian integer hinzufgen

- ► message in 512-bit chunks teilen
- ▶ foreach chunk{ teile chunk in sechzehn 32-bit big-endian Worte w[0..15]

ERWEITERUNG DER WORTE

```
for i = 16 to 63 { s0 := (w[i-15] \text{ rightrotate 7}) \text{ xor } (w[i-15] \text{ rightrotate 18})  xor (w[i-15] \text{ rightrotate 17}) \text{ xor } (w[i-2] \text{ rightrotate 17}) \text{ xor } (w[i-2] \text{ rightrotate 19}) \text{ xor } (w[i-2] \text{ rightshift 10}) w[i] := w[i-16] + s0 + w[i-7] + s1 }
```

IMPLEMENTIERUNG

Hashzuweisung

GESCHICHTE

$$a := h0$$

 $b := h1$
 $c := h2$
 $d := h3$
 $e := h4$
 $f := h5$
 $g := h6$

h := h7

Einführung

ANWENDUNG

AUSBLICK

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG ANWENDUNG
0 00000 00000000 00

HAUPTSCHLEIFE

for
$$i=0$$
 to 63 { $S0:=(a \text{ rightrotate 2}) \text{ xor } (a \text{ rightrotate 13}) \text{ xor } (a \text{ rightrotate 22})$

AUSBLICK

0

Secure Hash Algorithm 25 of 38

Implementierung ○○○○○○ Anwendung 00 AUSBLICK 0

HAUPTSCHLEIFE

for
$$i=0$$
 to 63 {
$$S0 := (a \text{ rightrotate 2}) \text{ xor } (a \text{ rightrotate } 13) \text{ xor } (a \text{ rightrotate } 22)$$

$$maj := (a \text{ and } b) \text{ xor } (a \text{ and } c) \text{ xor } (b \text{ and } c)$$

Secure Hash Algorithm 26 of 38

HAUPTSCHLEIFE

for
$$i=0$$
 to 63 {
$$S0 := (a \text{ rightrotate 2}) \text{ xor } (a \text{ rightrotate } 13) \text{ xor } (a \text{ rightrotate } 22)$$

$$maj := (a \text{ and } b) \text{ xor } (a \text{ and } c) \text{ xor } (b \text{ and } c)$$

$$t2 := S0 + maj$$

Secure Hash Algorithm 27 of 38

GESCHICHTE 00000

IMPLEMENTIERUNG

00000000

Anwendung oo

HAUPTSCHLEIFE

for
$$i=0$$
 to 63 {
$$S0 := (a \text{ rightrotate 2}) \text{ xor } (a \text{ rightrotate } 13) \text{ xor } (a \text{ rightrotate } 22)$$

$$maj := (a \text{ and } b) \text{ xor } (a \text{ and } c) \text{ xor } (b \text{ and } c)$$

$$t2 := S0 + maj$$

$$S1 := (e \text{ rightrotate } 6) \text{ xor } (e \text{ rightrotate } 11) \text{ xor } (e \text{ rightrotate } 25)$$

Secure Hash Algorithm 28 of 38

HAUPTSCHLEIFE

for
$$i=0$$
 to 63 {
$$S0 := (a \text{ rightrotate 2}) \text{ xor } (a \text{ rightrotate } 13) \text{ xor } (a \text{ rightrotate } 22)$$

$$maj := (a \text{ and } b) \text{ xor } (a \text{ and } c) \text{ xor } (b \text{ and } c)$$

$$t2 := S0 + maj$$

$$S1 := (e \text{ rightrotate } 6) \text{ xor } (e \text{ rightrotate } 11) \text{ xor } (e \text{ rightrotate } 25)$$

$$ch := (e \text{ and } f) \text{ xor } ((\text{not } e) \text{ and } g)$$

GESCHICHTE

HAUPTSCHLEIFE

for
$$i=0$$
 to 63 {
$$S0 := (a \text{ rightrotate } 2) \text{ xor } (a \text{ rightrotate } 13) \text{ xor } (a \text{ rightrotate } 22)$$

$$maj := (a \text{ and } b) \text{ xor } (a \text{ and } c) \text{ xor } (b \text{ and } c)$$

$$t2 := S0 + maj$$

$$S1 := (e \text{ rightrotate } 6) \text{ xor } (e \text{ rightrotate } 11) \text{ xor } (e \text{ rightrotate } 25)$$

$$ch := (e \text{ and } f) \text{ xor } ((\text{not } e) \text{ and } g)$$

$$t1 := h + S1 + ch + k[i] + w[i]$$

GESCHICHTE

ANWENDUNG

AUSBLICK

0

HAUPTSCHLEIFE

for
$$i=0$$
 to 63 {

 $S0:=(a \text{ rightrotate } 2) \text{ xor } (a \text{ rightrotate } 13) \text{ xor } (a \text{ rightrotate } 22)$
 $maj:=(a \text{ and } b) \text{ xor } (a \text{ and } c) \text{ xor } (b \text{ and } c)$
 $t2:=S0+maj$
 $S1:=(e \text{ rightrotate } 6) \text{ xor } (e \text{ rightrotate } 11) \text{ xor } (e \text{ rightrotate } 25)$
 $ch:=(e \text{ and } f) \text{ xor } ((\text{not } e) \text{ and } g)$
 $t1:=h+S1+ch+k[i]+w[i]$
 $h:=g$
 $g:=f$
 $f:=e$
 $e:=d+t1$
 $d:=c$
 $c:=b$
 $b:=a$
 $a:=t1+t2$

HAUPTSCHLEIFE

32 of 38

Secure Hash Algorithm

AUSGABE

digest = hash = h0 append h1 append h2 append h3 append h4 append h5 append h6 append h7

VERWENDUNGSZWECK

► Digitale Zertifikate und Signaturen

VERWENDUNGSZWECK

- ► Digitale Zertifikate und Signaturen
- ► Passwortverschlüsselung
 - ▶ pam_unix: sha2, md5
 - ► htpasswd(Apache): sha1, md5
 - ► MySQL: sha1



VERWENDUNGSZWECK

- ► Digitale Zertifikate und Signaturen
- ► Passwortverschlüsselung
 - ▶ pam_unix: sha2, md5
 - ► htpasswd(Apache): sha1, md5
 - ► MySQL: sha1
- ► Prüfsummen bei Downloads



Secure Hash Algorithm 36 of 38

- 1. sha2 fragil, kleine änderung im algo \rightarrow grosse auswirkung auf
- sicherheit 2. siehe quelle 14

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG ANWENDUNG AUSBLICK
0 00000 00000000 0 0 0

SICHERHEITSLÜCKEN & ANGRIFFSVEKTOREN



Secure Hash Algorithm 37 of 38

- 1. sha3 finalisten allesamt nicht von attacken gegen prinzipielles "merkle damgard "verfahren betroffen
- 2. merkle damgard: "Aus den Nachrichtenblöcken wird durch wiederholte Anwendung der Kompressionsfunktion der Hashwert erzeugt"

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG ANWENDUNG AUSBLICK

O 0000 00000000 00

SHA-3



Secure Hash Algorithm 38 o