Secure Hash Algorithm SHA-256

Chi Trung Nguyen *T-Systems*



21. Juni 2012



Einführung

AUSBLICK 00

- 1. um erinnerung etwas aufzufrischen: Was ist ein hash
- 2. geschichte gegliedert in einzelne mitglieder
- 3. implementierung sha256 und anwendungen von hashalgorithmen bzw sha2
- 4. schlussendlich kleiner ausblick auf sha3 der im sommer 2012 vorgestellt werden soll

AGENDA

EINFÜHRUNG Was ist ein Hash?



Secure Hash Algorithm 2 of

- 1. um erinnerung etwas aufzufrischen: Was ist ein hash
- 2. geschichte gegliedert in einzelne mitglieder
- 3. implementierung sha256 und anwendungen von hashalgorithmen bzw sha2
- 4. schlussendlich kleiner ausblick auf sha3 der im sommer 2012 vorgestellt werden soll

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG
0 00000 000000000

AGENDA

EINFÜHRUNG

Was ist ein Hash?

GESCHICHTE

SHA Allgemein

SHA-0

SHA-1

SHA-2



ANWENDUNG

AUSBLICK 00

Secure Hash Algorithm 3 o

- 1. um erinnerung etwas aufzufrischen: Was ist ein hash
- 2. geschichte gegliedert in einzelne mitglieder
- 3. implementierung sha256 und anwendungen von hashalgorithmen bzw sha2
- 4. schlussendlich kleiner ausblick auf sha3 der im sommer 2012 vorgestellt werden soll

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG ANWENDUNG
0 00000 00000000 00000

AGENDA

EINFÜHRUNG

Was ist ein Hash?

GESCHICHTE

SHA Allgemein

SHA-0

SHA-1

SHA-2

IMPLEMENTIERUNG

Algorithmus Pseudocode



AUSBLICK

Secure Hash Algorithm 4 c

- 1. um erinnerung etwas aufzufrischen: Was ist ein hash
- 2. geschichte gegliedert in einzelne mitglieder
- 3. implementierung sha256 und anwendungen von hashalgorithmen bzw sha2
- 4. schlussendlich kleiner ausblick auf sha3 der im sommer 2012 vorgestellt werden soll

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG
0 00000 000000000

AGENDA

EINFÜHRUNG

Was ist ein Hash?

GESCHICHTE

SHA Allgemein

SHA-0

SHA-1

SHA-2

IMPLEMENTIERUNG

Algorithmus

Pseudocode

ANWENDUNG

Verwendungszweck

Schwachstellen/Angriffsvektoren



ANWENDUNG

AUSBLICK

Secure Hash Algorithm 5 c

- 1. um erinnerung etwas aufzufrischen: Was ist ein hash
- 2. geschichte gegliedert in einzelne mitglieder
- 3. implementierung sha256 und anwendungen von hashalgorithmen bzw sha2
- 4. schlussendlich kleiner ausblick auf sha3 der im sommer 2012 vorgestellt werden soll

Einführung o GESCHICHTE 00000 IMPLEMENTIERUNG 00000000

Anwendung 00000 AUSBLICK 00

AGENDA

EINFÜHRUNG

Was ist ein Hash?

GESCHICHTE

SHA Allgemein

SHA-0

SHA-1

SHA-2

IMPLEMENTIERUNG

Algorithmus Pseudocode

ANWENDUNG

Verwendungszweck

Schwachstellen/Angriffsvektoren

AUSBLICK

SHA-3



Secure Hash Algorithm 6 of

- 1. bsp automarken, mercedes = 1,bmw = 2
- 2. krypto hash, hash unterschied: hash beliebig abgebildet(zb Perl), kryptohash = berechnet mit algorithmus

 EINFÜHRUNG
 GESCHICHTE
 IMPLEMENTIERUNG
 ANWENDUNG
 AUSBLICK

 ●
 00000
 00000000
 00000
 00
 00

WAS IST EIN HASH?

► deutsch: "zerhacken", "verstreuen"



Secure Hash Algorithm 70

- 1. bsp automarken, mercedes = 1,bmw = 2
- 2. krypto hash, hash unterschied: hash beliebig abgebildet(zb Perl), kryptohash = berechnet mit algorithmus

 EINFÜHRUNG
 GESCHICHTE
 IMPLEMENTIERUNG
 ANWENDUNG
 AUSBLICK

 ●
 00000
 000000000
 00000
 00
 00

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 8 c

- 1. bsp automarken, mercedes = 1,bmw = 2
- 2. krypto hash, hash unterschied: hash beliebig abgebildet(zb Perl), kryptohash = berechnet mit algorithmus

 EINFÜHRUNG
 GESCHICHTE
 IMPLEMENTIERUNG
 ANWENDUNG
 AUSBLICK

 ●
 00000
 000000000
 00000
 00
 00

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 9

- 1. sha-0 wurde...
- 2. sha steht für secure hash algorithm und ist eine gruppe von hash algorithmn

SHA ALLGEMEIN

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



Secure Hash Algorithm 10 of 48

- 1. sha-0 wurde...
- 2. sha steht für secure hash algorithm und ist eine gruppe von hash algorithmn

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 11 of 48

- ursprünglich als
 1991 empfohlen

Einführung GESCHICHTE IMPLEMENTIERUNG ANWENDUNG AUSBLICK 00000 00

SHA-0

► 1993 veröffentlicht



Secure Hash Algorithm 12 of 48

- 1. ursprünglich als
- 2. 1991 empfohlen

 EINFÜHRUNG
 GESCHICHTE
 IMPLEMENTIERUNG
 ANWENDUNG
 AUSBLICK

 0
 0 ●000
 000000000
 00000
 00

SHA-0

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



Secure Hash Algorithm 13 of 48

- 1. alles optional, nicht vorlesen!
- 2. 2005 von Xiaoyun Wang, Yiqun Lisa Yin und Hongbo Yu an der Shandong University in China gebrochen
- 3. Ihnen war es gelungen, den Aufwand zur Kollisionsberechnung von 2^{80} auf 2^{69} zu verringern
- 4. es wurde ein rechtsshift durch ein linksshift ersetzt
- 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

| IFÜHRUNG | GESCHICHTE | Implementierung | ANWENDUNG | AUSBLICK |
|----------|------------|-----------------|-----------|----------|
| | 00•00 | 00000000 | 00000 | 00 |

SHA-1

EIN

► 1995 veröffentlicht



Secure Hash Algorithm 14 of

- 1. alles optional, nicht vorlesen!
- 2. 2005 von Xiaoyun Wang, Yiqun Lisa Yin und Hongbo Yu an der Shandong University in China gebrochen
- 3. Ihnen war es gelungen, den Aufwand zur Kollisionsberechnung von 2^{80} auf 2^{69} zu verringern
- 4. es wurde ein rechtsshift durch ein linksshift ersetzt
- 5. 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

SHA-1

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



Secure Hash Algorithm 15 of

1. die hier gelistet sind(nächste folie)

Einführung o GESCHICHTE

IMPLEMENTIERUNG 00000000 Anwendung 00000 Ausblick 00

SHA-2

► 2002 veröffentlicht



Secure Hash Algorithm 16 of 48

1. die hier gelistet sind(nächste folie)

Einführung o GESCHICHTE

IMPLEMENTIERUNG

Anwendung 00000 AUSBLICK 00

SHA-2

- ► 2002 veröffentlicht
- ► existiert in mehreren Bit Variante



Secure Hash Algorithm 17 of 48

- 1. 2^{64} bits = 2 147 483 648 gigabytes
- 2. auch wenn 2¹²⁸ bits an daten möglich sind, werden 2⁶⁴ in der realität nicht überschritten
- 3. unterschiede erklären: andere konstanten bei 224 & 384, worte werden weggelassen

| ÜHRUNG | GESCHICHTE | Implementierung | ANWENDUNG | AUSBLICK |
|--------|------------|-----------------|-----------|----------|
| | 0000 | 00000000 | 00000 | 00 |

Tabelle: Secure Hash Algorithmus Eigenschaften

Einfü o

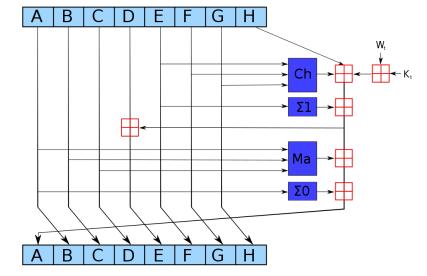
| Algorithmus | Message | Block Größe(bits) | Word Größe(bits) | Message Digest | |
|-------------|-------------|-------------------|------------------|----------------|--|
| | Größe(bits) | | | Größe(bits) | |
| SHA-1 | $< 2^{64}$ | 512 | 32 | 160 | |
| SHA-224 | $< 2^{64}$ | 512 | 32 | 224 | |
| SHA-256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



Secure Hash Algorithm 18 of 4

- 1. SHA2 teilt erst in messages,dann in 64 byte chunks, wendet mathematische funktionen auf jeden chunk, und fügt diese dann dem hash hinzu.
- 2. der algorithmus tut dies für jeden chunk und fügt es dann dem hash wert hinzu.
- 3. am ende bekommt man einen hash wert mit einer genauen länge

DARSTELLUNG DES ALGORITHMUS





Secure Hash Algorithm 19 of 4

GESCHICHTE 00000 IMPLEMENTIERUNG

•0000000

Anwendung 00000 Ausblick 00

FUNKTIONEN

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

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

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

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

PSEUDOCODE

► Initialisiere Variabeln (die ersten 32 Bits der Nachkommastellen der Quadratwurzeln von den ersten 8 Primzahlen 2..19):

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

PSEUDOCODE

► Initialisiere Variabeln (die ersten 32 Bits der Nachkommastellen der Quadratwurzeln von den ersten 8 Primzahlen 2..19):

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

► Initialisiere Variabeln der Runden Konstanten (die ersten 32 Bits der Nachkommastellen der Kubikwurzel von den ersten 64 Primzahlen 2..311):

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



- 1. bsp nachricht "abc"
- 2. representation in ascii: 97 = a, 98=b, 99=c
- 3. a= 0110 0001
- 4. b= 0110 0010
- 5. c= 0110 0011
- 6. da 8x3 = 24 ergibt sich 448 (24 + 1) = 423 zero Bits
- 7. am ende nachricht hinzufugen

Preprocessing

▶ bit 1 zur *message* hinzufügen



Secure Hash Algorithm 23 of 48

- 1. bsp nachricht "abc"
- 2. representation in ascii: 97 = a, 98=b, 99=c
- 3. a= 0110 0001
- 4. b= 0110 0010
- 5. $c = 0110\ 0011$
- 6. da 8x3 = 24 ergibt sich 448 (24 + 1) = 423 zero Bits
- 7. am ende nachricht hinzufugen

Preprocessing

- ▶ 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



Secure Hash Algorithm 24 of 48

- 1. bsp nachricht "abc"
- 2. representation in ascii: 97 = a, 98 = b, 99 = c
- 3. a= 0110 0001
- 4. b= 0110 0010
- 5. c= 0110 0011
- 6. da 8x3 = 24 ergibt sich 448 (24 + 1) = 423 zero Bits
- 7. am ende nachricht hinzufugen

Preprocessing

- ▶ 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



Secure Hash Algorithm 25 of 48

- 1. bsp nachricht "abc"
- 2. representation in ascii: 97 = a, 98=b, 99=c
- 3. a= 0110 0001
- 4. b= 0110 0010
- 5. $c = 0110\ 0011$
- 6. da 8x3 = 24 ergibt sich 448 (24 + 1) = 423 zero Bits
- 7. am ende nachricht hinzufugen

 EINFÜHRUNG
 GESCHICHTE
 IMPLEMENTIERUNG
 ANWENDUNG
 AUSBLICK

 0
 00000
 000000
 00000
 00000

Preprocessing

- ▶ 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]

64

423

Secure Hash Algorithm 26 of 48

ERWEITERUNG DER WORTE

```
for i=16 to 63 {
s0:=(w[i-15] \text{ rightrotate 7}) \text{ xor } (w[i-15] \text{ rightrotate 18})
\text{xor } (w[i-15] \text{ rightshift 3})
s1:=(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

000000000

GESCHICHTE

$$a := h0$$

$$b := h1$$

Einführung

$$c := h2$$

$$d := h3$$

$$e := h4$$

$$f := h5$$

$$g := h6$$
$$h := h7$$



28 of 48

ANWENDUNG

AUSBLICK

00

Einführung o GESCHICHTE 00000 IMPLEMENTIERUNG ○0000●000 Anwendung 00000 AUSBLICK 00

HAUPTSCHLEIFE

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

29 of 48

Secure Hash Algorithm

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 30 of 48

GESCHICHTE 00000 Implementierung ○0000●000 Anwendung 00000 AUSBLICK 00

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 31 of 48

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 32 of 48

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

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

$$t2 := S0 + maj$$

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

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

33 of 48

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

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

$$t2 := S0 + maj$$

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

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

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

AUSBLICK

00

ANWENDUNG

for
$$i=0$$
 to 63 {

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

GESCHICHTE

HAUPTSCHLEIFE

```
h0 := h0 + a
h1 := h1 + b
h2 := h2 + c
h3 := h3 + d
h4 := h4 + e
h5 := h5 + f
h6 := h6 + g
h7 := h7 + h
}
//Ende der foreach-Schleife
```

AUSGABE

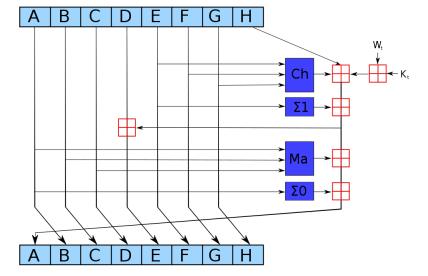
digest = hash = h0 append h1 append h2 append h3 append h4 append h5 append h6 append h7

- 1. SHA2 teilt erst in messages,dann in 64 byte chunks, wendet mathematische funktionen auf jeden chunk, und fügt diese dann dem hash hinzu.
- 2. der algorithmus tut dies für jeden chunk und fügt es dann dem hash wert hinzu.
- 3. am ende bekommt man einen hash wert mit einer genauen länge

EINFÜHRUNG GESCHICHTE **IMPLEMENTIERUNG** ANWENDUNG AUSBLICK

0 00000 0000000 00000 00000 00

DARSTELLUNG DES ALGORITHMUS





Secure Hash Algorithm 38 of 4

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



- 1. bis zum heutigen tage noch keine sicherheitslücken,daher eher schwachstellen als sicherheitslücken
- 2. allgemein wird in starke und schwache hashefunktionen unterschieden
- 3. wieder auf folie gucken
- 4. sha2 fragil, kleine änderung im algo \rightarrow grosse auswirkung auf sicherheit
- 5. siehe quelle 14

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG ANWENDUNG AUSBLICK
0 00000 00000000 00000 00000

SCHWACHSTELLEN/ANGRIFFSVEKTOREN

► Resistenzen:

- ► Kollisionsresistenz
- ► Preimage Resistenz
- ► Second Preimage Resistenz



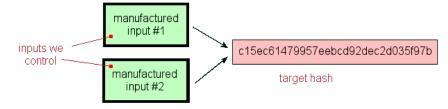
Secure Hash Algorithm 42 o

- 1. kollisionsfreiheit unmöglich, es muss nur schwer sein welche zu finden: da es unendlich viele eingaben, aber nur endlich viele hashes gibt
- 2. schwache kr: vorgegebene eingabe, gesucht ist eine eingabe, die zum gleichen hash führt
- 3. schwache kr: gesucht sind zwei beliebige eingaben, die zum gleichen hash führt

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG ANWENDUNG AUSBLICK
0 00000 00000000 00•00 00

KOLLISIONSRESISTENZ

Wie schwer ist es, zwei verschiedenen Nachrichten mit gleicher Prüfsumme zu finden?





Secure Hash Algorithm 43 of 48

PREIMAGE RESISTENZ

Wie schwer ist es, zu einem vorgegebenen Hash-Wert eine Nachricht zu erzeugen, die denselben Hash-Wert ergibt?

```
input we control input c693fc5c8c9b22deb6461ada7e2c3887
```

SECOND PREIMAGE RESISTENZ

Wie schwer ist es, zu einer vorgegebene Nachricht einen Hash-Wert eine Nachricht zu finden, die denselben Hash-Wert ergeben?

```
known original input hello, world input we control input we control input was input we control input we control input was inpu
```



Secure Hash Algorithm 45 of 48

- 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"
- 3. bisher langsamer, eventuell unnötig da sha2 ungebrochen + hoher migrationsaufwand

| HRUNG | GESCHICHTE | Implementierung | Anwendung | AUSBLICK |
|-------|------------|-----------------|-----------|----------|
| | 00000 | 00000000 | 00000 | •0 |

SHA-3

Einfüh

► 2007 rief NIST zu einem Wettbewerb auf



Secure Hash Algorithm 46 of

- 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"
- 3. bisher langsamer, eventuell unnötig da sha2 ungebrochen + hoher migrationsaufwand

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG ANWENDUNG AUSBLICK
0 00000 00000000 00000 •0

SHA-3

- ► 2007 rief NIST zu einem Wettbewerb auf
- ► 191 Einreichungen, 5 Finalisten



Secure Hash Algorithm 47 of

- 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"
- 3. bisher langsamer, eventuell unnötig da sha2 ungebrochen + hoher migrationsaufwand

EINFÜHRUNG GESCHICHTE IMPLEMENTIERUNG ANWENDUNG AUSBLICK
0 00000 00000000 00000 •0

SHA-3

- ▶ 2007 rief NIST zu einem Wettbewerb auf
- ► 191 Einreichungen, 5 Finalisten
- ► bisher langsamer als SHA2



Secure Hash Algorithm 48 of

IMPLEMENTIERUNG

Fragen?

GESCHICHTE

Einführung

AUSBLICK

ANWENDUNG