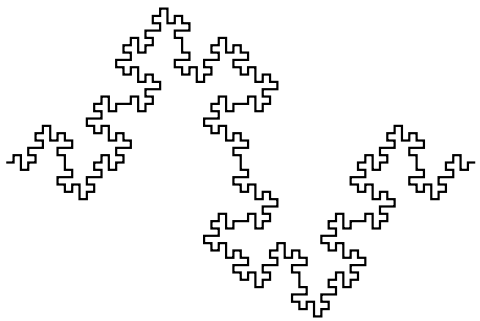
Secure Hash Algorithm

SHA-256

Chi Trung Nguyen
T-Systems



Agenda

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

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

SHA-0

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

SHA-1

- 1995 veröffentlicht
- aufgrund Designfehler in SHA-0

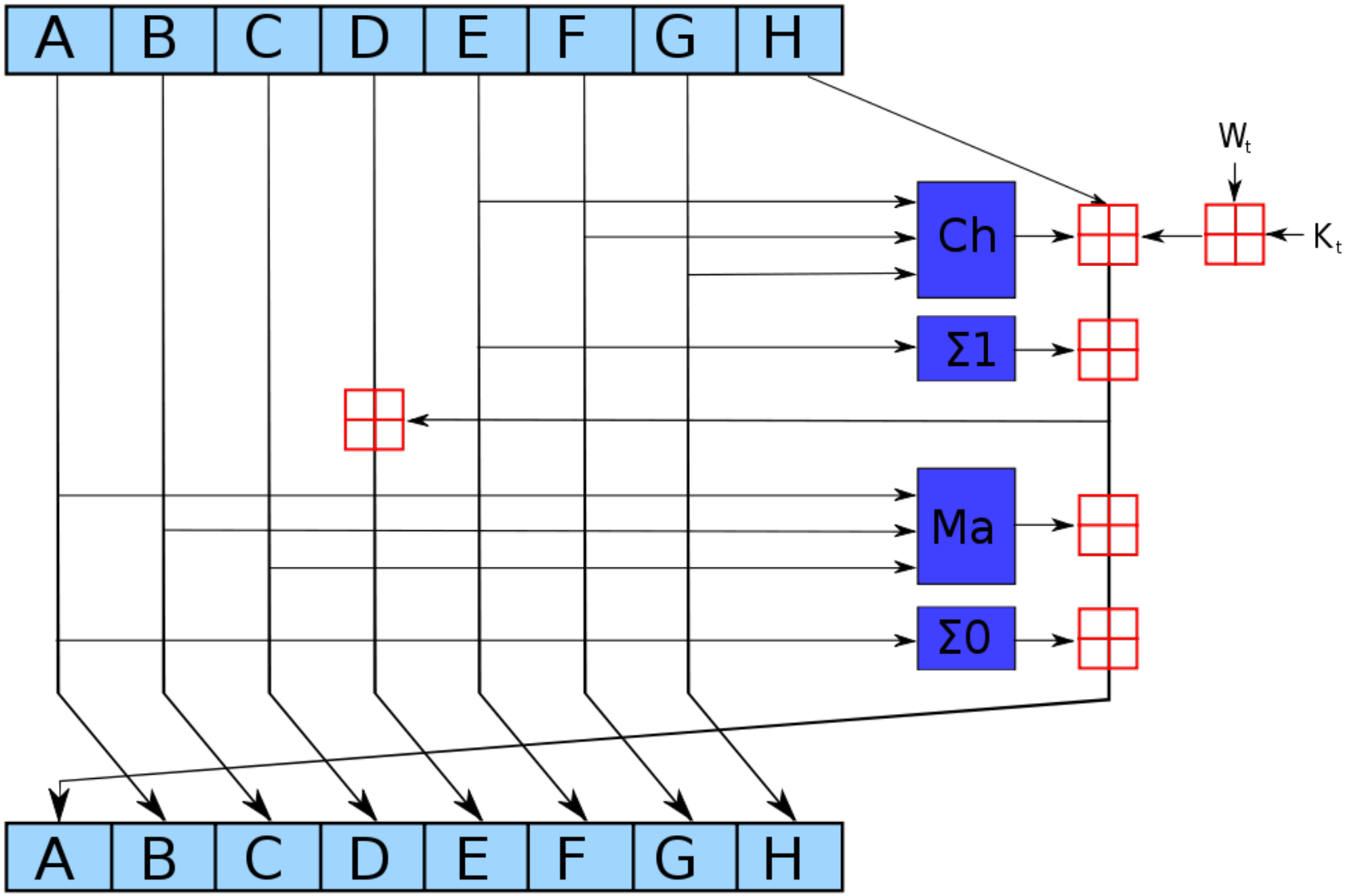
SHA-2

- 2002 veröffentlicht
- existiert in mehreren Bit Variante

Tabelle: Secure Hash Algorithmus Eigenschaften

Algorithmus	Message Größe(bits)	Block Größe(bits)	Word Größe(bits)	Message Digest 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

DARSTELLUNG DES ALGORITHMUS



FUNKTIONEN

$$Ch(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$Maj(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0 = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1 = (A \ggg 6) \oplus (A \ggg 11) \oplus (A \ggg 25)$$

PSEUDOCODE

- Initialisiere Variabeln (die ersten 32 Bits der Nachkommastellen der Quadratwurzeln von den ersten 8 Primzahlen 2..19):
- Initialisiere Variabeln der Runden Konstanten (die ersten 32 Bits der Nachkommastellen der Kubikwurzel von den ersten 64 Primzahlen 2..311):

PREPROCESSING

- $1message$
- $k0kmessage$
- $message$

- $message$
- $\{$
  $w[0..15]$

ERWEITERUNG DER WORTE

$i = 1663\{$

$s0$:= ($w[i-15]$ rightrotate 7) xor ($w[i-15]$ rightrotate 18) xor ($w[i-15]$ rightshift 3)

$s1$:= ($w[i-2]$ rightrotate 17) xor ($w[i-2]$ rightrotate 19) xor ($w[i-2]$ rightshift 10)

$w[i] := w[i-16] + s0 + w[i-7] + s1$

$\}$

HASHZUWEISUNG

$$
\begin{aligned}
a &:= h0 \\
b &:= h1 \\
c &:= h2 \\
d &:= h3 \\
e &:= h4 \\
f &:= h5 \\
g &:= h6 \\
h &:= h7
\end{aligned}
$$

Hauptschleife

$$
\begin{aligned}
i = 063\{ \\
& S0a2a13a22 \\
& majabacbc \\
& t2S0maj \\
& S1e6e11e25 \\
& chefeg \\
& t1hS1chk[i]w[i] \\
& h := g \\
& g := f \\
& f := e \\
& e := d + t1 \\
& d := c \\
& c := b \\
& b := a \\
& a := t1 + t2
\end{aligned}
$$

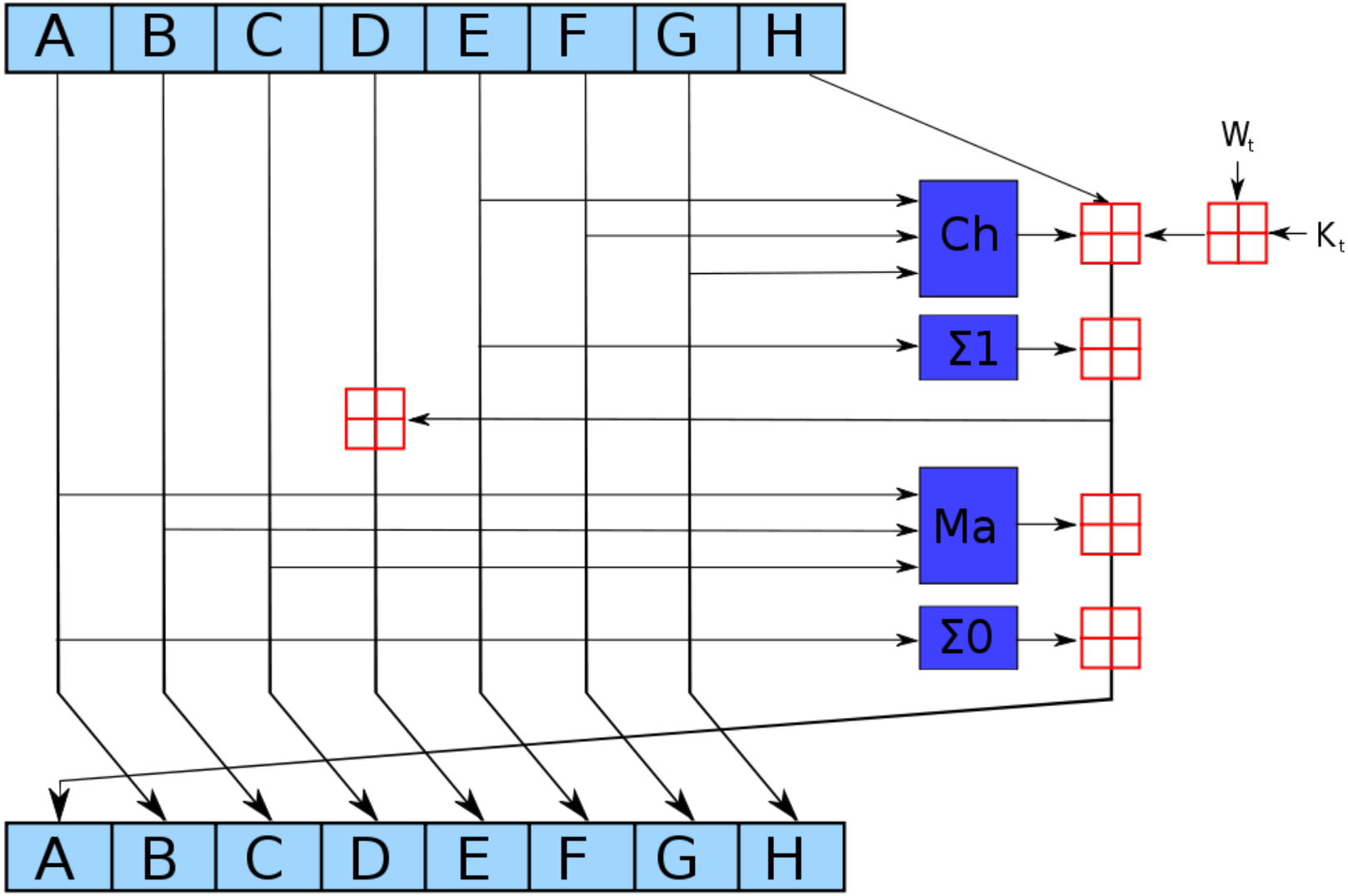
HAUPTSCHLEIFE

*h*0*h*0*a*
*h*1*h*1*b*
*h*2*h*2*c*
*h*3*h*3*d*
*h*4*h*4*e*
*h*5*h*5*f*
*h*6*h*6*g*
*h*7*h*7*h*
}
} *//Ende der foreach-Schleife*

AUSGABE

$$h0h1h2h3h4h5h6h7$$

DARSTELLUNG DES ALGORITHMUS



VERWENDUNGSZWECK

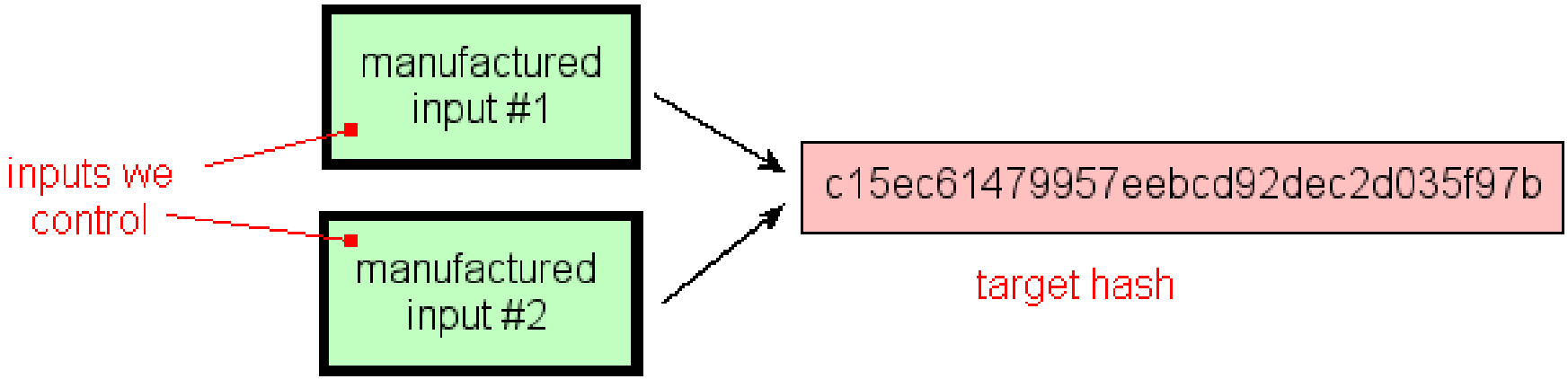
- Digitale Zertifikate und Signaturen
- Passwortverschlüsselung
  - pam_unix: sha2, md5
  - htpasswd(Apache): sha1, md5
  - MySQL: sha1
- Prüfsummen bei Downloads

SCHWACHSTELLEN / ANGRIFFSVEKTOREN

- Resistenzen:
  - Kollisionsresistenz
  - Preimage Resistenz
  - Second Preimage Resistenz

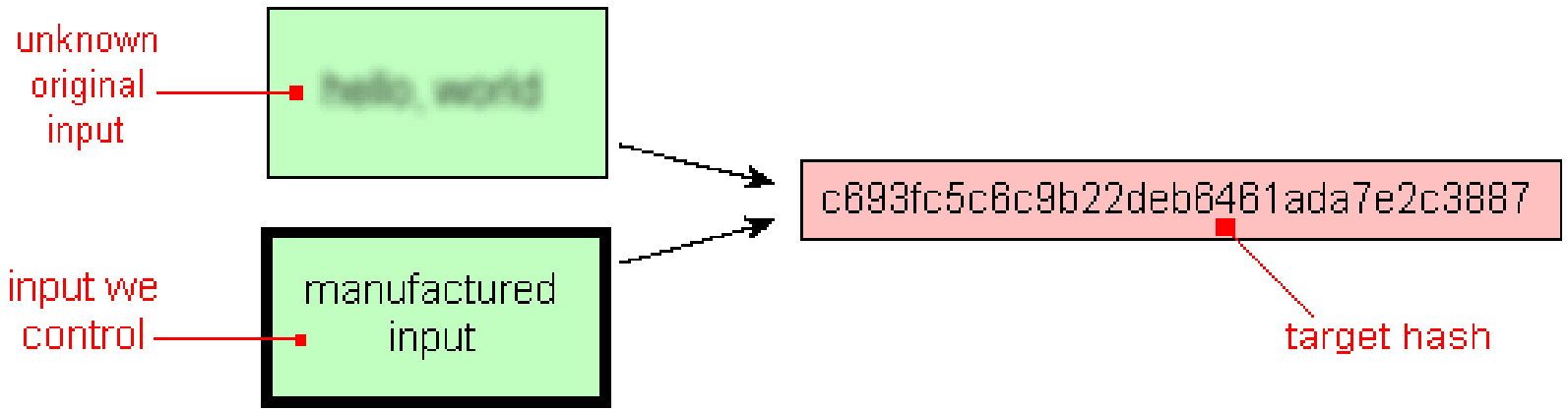
KOLLISIONSRESISTENZ

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



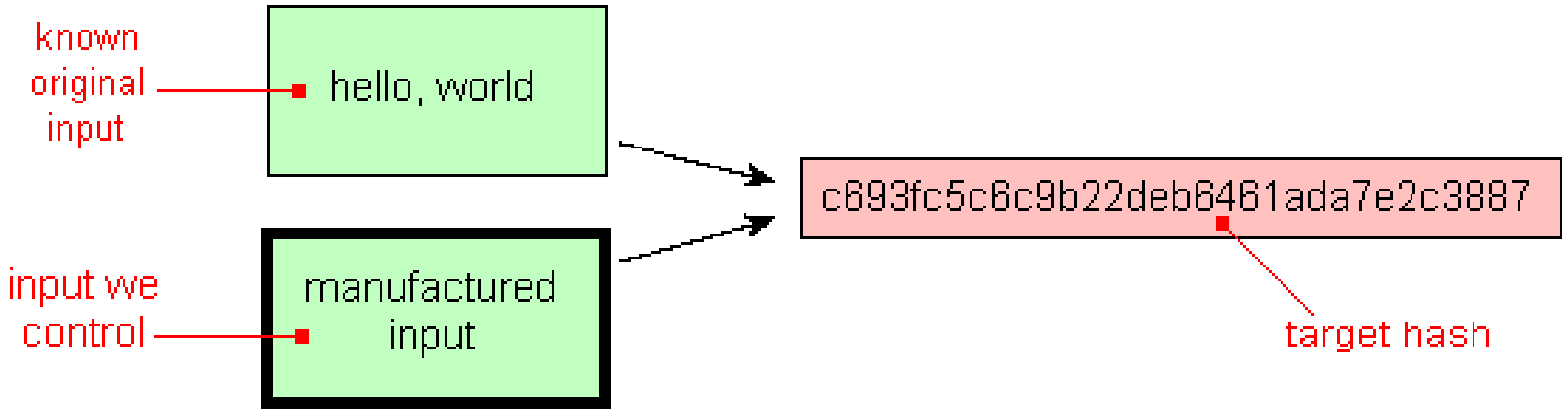
PREIMAGE RESISTENZ

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



SECOND PREIMAGE RESISTENZ

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



SHA-3

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

Fragen?