Institut für Theoretische Informatik
Prof. Dr. Dennis Hofheinz
Übungsleiter: Thomas Agrikola

Stammvorlesung Sicherheit im Sommersemester 2017

Übungsblatt 3

Aufgabe 1. Beurteilen Sie für die folgenden Konstruktionen jeweils, ob es sich für beliebige kollisionsresistente Hashfunktionen $H, H' : \{0,1\}^* \to \{0,1\}^k$ ebenfalls um kollisionsresistente Hashfunktionen handelt. Falls ja, beweisen Sie diese Aussage indem Sie zeigen, wie sich aus einer Kollision für $\hat{H}$ eine Kollision für H oder H' entwickeln ließe. Falls nicht, geben Sie konkrete Wahlen für H und H' und eine Kollision an.

(a) $\hat{H}(x) := H(x\|x)$

(b) $\hat{H}(x) := H(x) \oplus H'(x)$

(c) $\hat{H}(x) := H(H'(x))$

(d) $\hat{H}(x) := F(x)\|H(x)$ für eine beliebige Funktion $F : \{0,1\}^* \to \{0,1\}^k$

(e) $\hat{H}(x) := \begin{cases} x & \text{wenn } |x| = k \\ H(x) & \text{sonst} \end{cases}$

(f) $\hat{H}(x) := H(F(x))$ für eine beliebige injektive Funktion $F : \{0,1\}^* \to \{0,1\}^*$

Dabei bezeichne $\|$ die Konkatenation von Bitstrings und $\oplus$ Verknüpfung durch bitweises XOR.

Lösungsvorschlag zu Aufgabe 1.

(a) $\hat{H}(x) := H(x\|x)$ ist kollisionsresistent: Sei $\hat{H}(x) = \hat{H}(y)$ eine Kollision (also $x \neq y$) für $\hat{H}$, so ist $H(x\|x) = H(y\|y)$ eine Kollision für H (da $x \neq y \Rightarrow x\|x \neq y\|y$). Da H kollisionsresistent ist, dürfen auf Kollisionen für $\hat{H}$ nur schwer zu finden sein. Also ist $\hat{H}$ kollisionsresistent.

(b) $\hat{H}(x) := H(x) \oplus H'(x)$ ist im Allgemeinen nicht kollisionsresistent. Z.B. für $H = H'$ ist $\hat{H}(x) = 0^k$ für alle x.

(c) $\hat{H}(x) := H(H'(x))$ ist kollisionsresistent. Sei $\hat{H}(x) = \hat{H}(y)$ also $H(H'(x)) = H(H'(y))$. Ist $H'(x) \neq H'(y)$, so haben wir eine Kollision für H gefunden. Ansonsten ist $H'(x) = H'(y)$ eine Kollision für H' (da $x \neq y$).

(d) $\hat{H}(x) := F(x)\|H(x)$ für eine beliebige Funktion $F : \{0,1\}^* \to \{0,1\}^k$ ist kollisionsresistent. Da $\hat{H}(x) = \hat{H}(y)$ insbesondere $H(x) = H(y)$ impliziert (die hinteren Teile der Ausgabe von $\hat{H}$ müssen übereinstimmen) liefert uns eine Kollision für $\hat{H}$ unmittelbar eine Kollision für H.

(e) $\hat{H}(x) := \begin{cases} x & \text{wenn } |x| = k \\ H(x) & \text{sonst} \end{cases}$ ist nicht kollisionsresistent. Für alle y mit $|y| \neq k$ ist $\hat{H}(y) = H(y)$. Da $|H(y)| = k$ ist aber auch $\hat{H}(H(y)) = H(y)$. Somit ist also $y, H(y)$ eine Kollision für $\hat{H}$, da auch $y \neq H(y)$ gilt (das folgt direkt aus der unterschiedlichen Bitlänge).

(f) $\hat{H}(x) := H(F(x))$ für eine beliebige injektive Funktion $F : \{0,1\}^* \to \{0,1\}^*$ ist kollisionsresistent. Sei $\hat{H}(x) = \hat{H}(y)$ (für $x \neq y$) und damit $H(F(x)) = H(F(y))$. Da F injektiv gilt $F(x) \neq F(y)$ und wir haben eine Kollision für H gefunden.

Aufgabe 2.

(a) Im ersten Teil der RSA-Schlüsselgenerierung wurden für einen Benutzer A die Primzahlen $P = 23$ und $Q = 11$ gezogen.

(i) Setzen Sie die RSA-Schlüsselgenerierung fort. Berechnen Sie einen zu P und Q gehörigen öffentlichen RSA-Schlüssel $pk_A = (N, e)$ und einen privaten RSA-Schlüssel $sk_A = (N, d)$.
Hinweis: Führen Sie den erweiterten euklidischen Algorithmus zur Übung von Hand durch. Sollten Sie den erweiterten euklidischen Algorithmus nicht kennen, so finden Sie im Kapitel 5 des Skripts eine Beschreibung des Algorithmus.

(ii) Senden Sie die Nachricht $M = 17$ RSA-verschlüsselt an Benutzer A. Benutzen Sie dazu die Lehrbuch-Variante von RSA (ohne Padding) und den öffentlichen Schlüssel aus (a).

(iii) Nehmen wir an, Sie seien Benutzer A. Entschlüsseln Sie das empfangene Chiffrat aus (b).

(b) Bei der Rabin-Variante der RSA-Einwegfunktion wird als öffentlicher Schlüssel immer $e = 2$ gewählt, es ist also $\mathsf{Rab}_N(x) = x^2 \bmod N$, wobei $N = P \cdot Q$ ein RSA-Modulus ist. Warum lässt sich die Rabin-Funktion nicht genauso wie die RSA-Funktion durch Potenzieren mit einem geheimen Schlüssel d invertieren?

(c) Angenommen es gäbe einen effizienten (d. h. die Laufzeit ist polynomiell in der Eingabelänge) Algorithmus $\mathcal{A}$ welcher die Rabin-Funktion invertiert, also für gegebenes N und y einen Wert x berechnet mit $\mathsf{Rab}_N(x) = y$. Zeigen Sie: Dann gibt es auch einen effizienten Algorithmus $\mathcal{B}$ welcher RSA-Moduli N zu P und Q faktorisiert.
Hinweise: Gemäß der dritten binomischen Formel $a^2 - b^2 = (a+b) \cdot (a-b)$ liefert jede quadratische Kongruenz $a^2 = b^2 \bmod N$ eine Faktorisierung der Null: $(a + b) \cdot (a - b) = 0 \bmod N$. Ferner gilt, wenn N ein RSA-Modulus ist, dass die meisten Quadrate in $\mathbb{Z}_N$ vier verschiedene Quadratwurzeln haben.

Lösungsvorschlag zu Aufgabe 2.

(a)

(i) $P = 23$ und $Q = 11$ führen zu $N = P \cdot Q = 253$ als RSA-Modulus. Während der Schlüsselgenerierung sind P und Q und damit insbesondere $\varphi(N) = (P - 1)(Q - 1) = 220$ (φ ist hier die eulersche φ-Funktion; $\varphi(N)$ die Ordnung der multiplikativen Gruppe $\mathbb{Z}_N = \mathbb{Z}/N\mathbb{Z}$). Wir führen die RSA-Schlüsselgenerierung aus der Vorlesung fort.

- Wir ziehen gleichverteilt $e \leftarrow \{3, \ldots, \varphi(N) - 1\}$
- ... bis $\mathsf{gcd}(e, \varphi(N)) = 1$ gilt. In unserem Fall wurde $e := 19$ gezogen.
- Wir berechnen $d = e^{-1} \bmod \varphi(N)$ mit erweitertem Euklid für $e = 19$ und $\varphi(N) = 220$:

(i) $220 = 11 \cdot 19 + 11$
(ii) $19 = 1 \cdot 11 + 8$
(iii) $11 = 1 \cdot 8 + 3$
(iv) $8 = 2 \cdot 3 + 2$
(v) $3 = 1 \cdot 2 + 1$
(vi) $2 = 2 \cdot 1$

Nun zurück: $1 = 3 - 1 \cdot 2$ (wg. (v))
$1 = 3 - 1 \cdot (8 - 2 \cdot 3) = -8 + 3 \cdot 3$ (wg. (iv), 2 ersetzen)
$1 = -8 + 3 \cdot (11 - 1 \cdot 8) = 3 \cdot 11 - 4 \cdot 8$ (wg. (iii), 3 ersetzen)
$1 = 3 \cdot 11 - 4 \cdot (19 - 1 \cdot 11) = -4 \cdot 19 + 7 \cdot 11$ (wg. (ii), 8 ersetzen)
$1 = -4 \cdot 19 + 7 \cdot (220 - 11 \cdot 19) = 7 \cdot 220 - 81 \cdot 19$ (wg. (i), 11 ersetzen)
Das inverse Element zu $e = 19$ im Exponenten (wo wir modulo $\varphi(N) = 220$ rechnen) ist also $d = -81 (\equiv 139 \bmod 220)$. Wir erhalten somit $pk_A := (N, e) = (253, 19)$ als öffentlichen Schlüssel und $sk_A := (N, d) = (253, 139)$ als geheimen Schlüssel für Benutzer A.
Hinweis: Man kann die Koeffizienten 7 und -81 für die Gleichung $1 = 7 \cdot 220 - 81 \cdot 19$ auch direkt auf dem „Hinweg“ mit ausrechnen und spart sich so den „Rückweg“.

(ii) Wir verschlüsseln die Nachricht $M = 17$ mittels des öffentlichen Schlüssels $pk_A = (N, e) = (253, 19)$ aus (a) zu $C_A := M^e \bmod N = 17^{19} \bmod 253 = 189$.

(iii) Die Entschlüsselung mittels des geheimen Schlüssels $sk_A = (N, d) = (253, 139)$ von Benutzer A ergibt $M := C_A^d \bmod N = 189^{139} \bmod 253 = 17$.

(b) Der Schlüssel $e = 2$ hat kein Inverses modulo $\varphi(N)$; es gilt $\mathrm{ggT}\big(2, \varphi(N)\big) = \mathrm{ggT}\big(2, (P-1)(Q-1)\big) = 2$.

(c) Gegeben eine Eingabe N, läuft $\mathcal{B}$ folgendermaßen ab:

(1.) Ziehe ein zufälliges $x \in \mathbb{Z}_N$ und berechne $y = x^2 \bmod N$. Dann lasse $\mathcal{A}$ mit Eingabe (N, y) laufen. (Falls y vier verschiedene Quadratwurzeln hat, gilt für die Ausgabe x' von $\mathcal{A}$ mit Wahrscheinlichkeit $\frac{1}{2}$ weder $x' = x \bmod N$ noch $x' = -x \bmod N$. Auf jeden Fall gilt immer $(x + x')(x - x') = 0 \bmod N$.)

(2.) Falls $x' \in \{x, -x\} \bmod N$, gehe zurück zu (1.). Andernfalls setze $P := \mathrm{ggT}(x + x', N)$ und berechne $Q = N/P$. (Die Werte $x + x'$ und $x - x'$ sind jeweils keine Vielfache von N, ihr Produkt aber schon. Sie müssen also jeweils genau einen Primfaktor von N enthalten.)

Aufgabe 3. Aus der Vorlesung ist bekannt, dass mithilfe einer kollisionsresistenten Kompressionsfunktion $\mathsf{F} : \{0,1\}^{2k} \to \{0,1\}^k$, mit $k \in \mathbb{N}$, eine kollisionsresistente Hashfunktion H_{MD} mittels des Merkle-Damgård-Verfahrens konstruiert werden kann. Eine mögliche Realisierung funkioniert wie folgt: Bei Eingabe von $M \in \{0,1\}^*$, mit $|M| = L < 2^k$, führe folgenden Algorithmus aus:

1. Setze $B := \lceil \frac{L}{k} \rceil$. ($B$ entspricht der Anzahl der Blöcke von M der Länge k.) Füge an das Ende der Nachricht M so viele Nullen an bis die Länge ein Vielfaches von k ergibt. Somit kann die Nachricht in die einzelnen k-Bit-Blöcke $M_1, \ldots, M_B$ aufgeteilt werden.
2. Setze $M_{B+1} := L$, wobei L mit exakt k Bits codiert wird.
3. Setze $Z_0 := 0^k$.
4. Für $i = 1, \ldots, B + 1$ berechne $Z_i := \mathsf{F}(Z_{i-1} \| M_i)$.
5. Gib Z_{B+1} aus.

Somit erhalten wir den Hashwert $\mathsf{H}_{\mathsf{MD}}(M) := Z_{B+1}$. Die Länge der Nachricht fließt also in den Hashwert ein. Sei nun $\mathsf{H}'_{\mathsf{MD}}$ eine Variante, die Z_B statt Z_{B+1} ausgibt. Geben Sie eine Kollision für $\mathsf{H}'_{\mathsf{MD}}$ an.

Betrachten wir nun nur Nachrichten, deren Länge ein Vielfaches der Blocklänge k ist. Lässt sich die Sicherheit von $\mathsf{H}'_{\mathsf{MD}}$ unter dieser Einschränkung für alle kollisionsresistenten Kompressionsfunktion F beweisen?

Lösungsvorschlag zu Aufgabe 3.

- $|M|$ **beliebig:** Fließt die Länge der Nachricht nicht in den Hashwert ein, lassen sich leicht Kollisionen finden, indem man einen Teil des Paddings der Nachricht zuordnet. Sei beispielsweise $M \in \{0,1\}^*$, mit $|M| \bmod k \neq 0$. Dann gilt $\mathsf{H}'_{\mathsf{MD}}(M) = \mathsf{H}'_{\mathsf{MD}}(M \| 0^\ell)$ für jedes $\ell \in \{0, \ldots, k - |M| \bmod k\}$. Für alle $\ell \neq 0$ ist dies eine Kollision.
- $|M|$ **Vielfaches der Blocklänge** k**:** Nein, auch auf der Menge der Nachrichten, deren Länge ein Vielfaches der Blocklänge k ist, lassen sich Kollisionen finden. Um dieses Problem zu illustrieren, konstruieren wir uns zunächst eine spezielle Kompressionsfunktion F_M. Es sei $F : \{0,1\}^{2k} \to \{0,1\}^{k-1}$ eine kollisionsresistente Kompressionsfunktion und $M \in \{0,1\}^k$ beliebig. Wir setzen

$$F_M(x) := \begin{cases} 0^k & \text{, wenn } x = 0^k \| M \\ F(x) \| 1 & \text{, sonst} \end{cases}$$

Behauptung: F_M ist kollisionsresistent.
Beweis: Sei $F_M(x) = F_M(y)$ für $x \neq y$ eine Kollision für F_M. Ist $x \neq 0^k \| M$ und $y \neq 0^k \| M$ so haben wir eine Kollision für F gefunden, denn es gilt:

$$F_M(x) = F(x) \| 1 = F(y) \| 1 = F_M(y) \Rightarrow F(x) = F(y).$$

F ist aber kollisionsresistent. Sei also o.B.d.A. $x = 0^k \| M$. Dann gilt:

$$F_M(x) = 0^k = F(y) \| 1 = F_M(y).$$

Auch das kann allerdings nicht sein, da $0 \neq 1$ (man betrachte das niederwertigste Bit von 0^k und $F(y)\|1$. Somit ist F_M kollisionsresistent. $\square$

Für die aus F_M nach dem Merkle-Damgård-Verfahren konstruierte Hashfunktion H_{MD} gilt bei der Berechnung von $\mathsf{H}_{\mathsf{MD}}(M^a)$ (für $a \in \mathbb{N}$):

$$\begin{aligned}
Z_0 &= 0^k \\
Z_1 &= F_M(Z_0 \| M) &&= F_M(0^k \| M) = 0^k \\
Z_2 &= F_M(Z_1 \| M) &&= F_M(0^k \| M) = 0^k \\
&\vdots && \vdots \\
Z_B &= F_M(Z_{B-1} \| M) &&= F_M(0^k \| M) = 0^k
\end{aligned}$$

D.h. es gilt $\mathsf{H}_{\mathsf{MD}}(M^a) = 0^k$. Somit gilt aber auch für $a, b \in \mathbb{N}$, $a \neq b$, dass

$$\mathsf{H}_{\mathsf{MD}}(M^a) = 0^k = \mathsf{H}_{\mathsf{MD}}(M^b) \wedge M^a \neq M^b,$$

womit wir beliebig viele Kollisionen gefunden haben. Die so konstruierte Hashfunktion ist also nicht kollisionsresistent, obwohl wir uns auf Nachrichten deren Bitlänge ein Vielfaches der Blocklänge k ist, beschränkt haben.

Aufgabe 4. Der ebenso geniale wie geldgierige Wissenschaftler und Superbösewicht Doktor Meta ist pleite. Um sein Imperium wieder aufbauen zu können, benötigt er viel Kapital, und das schnell. Eine naheliegende Lösung ist die Einführung einer Kryptowährung MetaCoin, in der er seine Handlanger auszahlen kann.

MetaCoin funktioniert folgendermaßen:

- Doktor Meta gibt eine Hashfunktion H öffentlich bekannt.
- Ein MetaCoin ist äquivalent zu einer Kollision der Hashfunktion, d.h. um einen MetaCoin zu erhalten, muss man eine Kollision berechnen.
- Gefundene Kollisionen werden für alle zugänglich veröffentlicht (natürlich so, dass niemand bereits veröffentlichte Kollisionen mehrmals verwenden kann – das soll hier aber nicht weiter vertieft werden).

Damit Doktor Meta sich selbst beliebig viele MetaCoins berechnen kann, konstruiert er die Hashfunktion so, dass er selbst in der Lage ist, einfach Kollisionen zu berechnen. Sei dazu $\mathbb{G}$ eine zyklische Gruppe mit Erzeuger g und primer Ordnung p. Seine Hashfunktion sieht wie folgt aus:

- Doktor Meta zieht zufällig $x \leftarrow \mathbb{Z}_p$ und berechnet $h := g^x$.
- Jede Nachricht M ist ein Tupel (M_1, M_2) aus $\mathbb{Z}_p \times \mathbb{Z}_p$.
- Die Hashfunktion wird folgendermaßen ausgewertet:

$$H(M) = H((M_1, M_2)) = g^{M_1} h^{M_2} \qquad (= g^{M_1}(g^x)^{M_2} = g^{M_1 + xM_2}).$$

- g und h sind öffentlich bekannt (damit jeder die Hashfunktion auswerten kann), x ist geheim und nur Doktor Meta bekannt.

(a) Zeigen Sie, dass H kollisionsresistent ist. Gehen Sie dazu davon aus, dass es schwierig ist in $\mathbb{G}$ diskrete Logarithmen zu ziehen, d.h. für alle PPT-Angreifer $\mathcal{A}$ ist die Wahrscheinlichkeit

$$\Pr[\mathcal{A}(g, g^x) = x \mid x \leftarrow \mathbb{Z}_p]$$

vernachlässigbar.

Hinweis: Zeigen Sie dazu, wie aus einem PPT-Angreifer $\mathcal{A}$, der effizient Kollisionen berechnet, ein PPT-Angreifer $\mathcal{B}$ konstruiert werden kann, der effizient diskrete Logarithmen in $\mathbb{G}$ zieht.

(b) Warum (und wie) kann Doktor Meta trotzdem effizient Kollisionen berechnen?

(c) Doktor Meta ist bei der Konstruktion der Hashfunktion ein schwerwiegender Fehler unterlaufen. Sobald auch nur eine einzige Kollision öffentlich bekannt wird, kann jeder effizient Kollisionen berechnen. Warum?

Lösungsvorschlag zu Aufgabe 4.

(a) Wir nehmen zum Widerspruch an, dass H nicht kollisionsresistent ist, d.h. es existiert ein PPT-Algorithmus $\mathcal{A}$, der mit nicht-vernachlässigbarer Wahrscheinlichkeit Kollisionen für H berechnet. Wir konstruieren daraus einen PPT-Algorithmus $\mathcal{B}$, der diskrete Logarithmen in $\mathbb{G}$ berechnen kann.

$\mathcal{B}$ erhält als Eingabe g, g^x für ein zufälliges $x \in \mathbb{Z}_p$ (Achtung, $\mathcal{B}$ kennt x nicht, sondern soll x berechnen!). Damit konstruiert er die Hashfunktion H wie vorgeschrieben und gibt diese an $\mathcal{A}$ weiter, d.h. er übergibt g und $h := g^x$ an $\mathcal{A}$. $\mathcal{A}$ gibt nun mit nicht-vernachlässigbarer Wahrscheinlichkeit eine Kollision aus. Er berechnet also zwei Nachrichten $M = (M_1, M_2), M^* = (M_1^*, M_2^*) \in \mathbb{Z}_p$ mit $M \neq M^*$. Nun gilt:

$$\begin{aligned} H(M) = H(M^*) &\Leftrightarrow \quad g^{M_1} h^{M_2} = g^{M_1^*} h^{M_2^*} \\ &\Leftrightarrow \quad g^{M_1 + xM_2} = g^{M_1^* + xM_2^*} \\ &\Leftrightarrow M_1 + xM_2 = M_1^* + xM_2^* \\ &\Leftrightarrow \qquad x = \frac{M_1^* - M_1}{M_2 - M_2^*}. \end{aligned}$$

Es gilt noch zu zeigen, dass $M_2 - M_2^* \neq 0$. Nehmen wir dazu an, dies wäre nicht der Fall und es ist $M_2 = M_2^*$. Dann würde gelten

$$g^{M_1} h^{M_2} = g^{M_1^*} h^{M_2^*} \Leftrightarrow g^{M_1} = g^{M_1^*} \Leftrightarrow M_1 = M_1^*,$$

was im Widerspruch zu $M \neq M^*$ steht (analog kann man zeigen, dass $M_2 \neq M_2^*$).

Somit kann $\mathcal{B}$ mithilfe einer Kollision sehr einfach das gesuchte x berechnen. Dies steht im Widerspruch zur Annahme, dass der diskrete Logarithmus in $\mathbb{G}$ schwierig ist. Somit muss also auch die Wahrscheinlichkeit, dass $\mathcal{A}$ eine Kollision berechnet, vernachlässigbar sein, woraus die Kollisionsresistenz von H folgt.

(b) Da Doktor Meta das x selbst zieht, kennt er dieses natürlich. Mithilfe von x kann er sich folgendermaßen beliebige Kollisionen berechnen:

- Er zieht sich drei beliebige Werte $M_1, M_2, M_1^* \leftarrow \mathbb{Z}_p$ mit $M_1 \neq M_1^*$.
- Um M_2^* zu bestimmen, löst er die Gleichung:

$$\begin{aligned} g^{M_1} h^{M_2} = g^{M_1^*} h^{M_2^*} &\Leftrightarrow \quad g^{M_1 + xM_2} = g^{M_1^* + xM_2^*} \\ &\Leftrightarrow M_1 + xM_2 = M_1^* + xM_2^* \\ &\Leftrightarrow \qquad M_2^* = \frac{M_1 - M_1^*}{x} + M_2 \end{aligned}$$

- Er setzt $M = (M_1, M_2)$ und $M^* = (M_1^*, M_2^*)$ (wie in (a) gilt $M_2 \neq M_2^*$, da $M_1 \neq M_1^*$).

M, M^* ist eine gültige Kollision, wie sich durch einsetzen leicht nachrechnen lässt.

(c) Wird eine Kollision öffentlich bekannt, kann jeder mit dem Algorithmus aus Teilaufgabe (a) x berechnen. Sobald man aber x kennt, kann man das Vorgehen aus Teilaufgabe (b) verwenden, um beliebige Kollisionen zu berechnen.