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.

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.

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?

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?