Karlsruher Institut für Technologie

Institut für Theoretische Informatik
Prof. Dr. D. Hofheinz

Stammvorlesung Sicherheit im Sommersemester 2017

Nachklausur

Lösung
06.10.2017

Vorname: _____

Nachname: _____

Matrikelnummer: _____

Klausur-ID: _____

Hinweise

- Schreiben Sie auf **alle Blätter** der Klausur und **alle Zusatzblätter** Ihre Matrikelnummer und Ihren Namen.
- Für die Bearbeitung stehen Ihnen 60 Minuten zur Verfügung.
- Es sind keine Hilfsmittel zugelassen.
- Schreiben Sie Ihre Lösungen auf die Aufgabenblätter sowie auf deren Rückseiten.
- Bitte kennzeichnen Sie deutlich, welche Lösung gewertet werden soll. Bei mehreren angegebenen Möglichkeiten wird jeweils die schlechteste Alternative gewertet.
- Zusätzliches Papier erhalten Sie bei Bedarf von der Aufsicht.
- Die Klausur umfasst 10 Seiten.

Aufgabe	mögliche Punkte					erreichte Punkte				
	a	b	c	d	Σ	a	b	c	d	Σ
1	5	6	-	-	11			-	-	
2	4	3	1	2	10					
3	10	2	2	-	14				-	
4	1	3	2	5	11					
5	3	6	5	-	14				-	
Σ					60					

Aufgabe 1. (5 + 6 Punkte)

Wir betrachten das RSA-Verschlüsselungsverfahren aus der Vorlesung (ohne Padding).

(a) Seien $P = 17$, $Q = 23$ und $N = P \cdot Q = 391$. Ferner sei der öffentliche Exponent $e = 61$ gewählt. Berechnen Sie den zugehörigen geheimen Exponenten d. Geben Sie alle Zwischenschritte an.

(b) Sei $N = P \cdot Q$ ein RSA-Modulus für ungerade, hinreichend große Primzahlen $P \neq Q$. Beweisen Sie folgende Aussage:

> Falls ein PPT-Algorithmus $\mathcal{A}$ existiert, der bei Eingabe N mit nicht-vernachlässigbarer Wahrscheinlichkeit $\epsilon_{\mathcal{A}}$ den Wert $\varphi(N)$ berechnet, dann existiert ein PPT-Algorithmus $\mathcal{B}$, der bei Eingabe N mit nicht-vernachlässigbarer Wahrscheinlichkeit $\epsilon_{\mathcal{B}}$ die Faktorisierung von N berechnet.

Geben Sie dazu insbesondere die Erfolgswahrscheinlichkeit ϵ_B an und begründen Sie, wieso diese nicht-vernachlässigbar ist. Argumentieren Sie zudem, wieso die Laufzeit des Algorithmus $\mathcal{B}$ polynomiell ist.

Lösungsvorschlag zu Aufgabe 1.

(a) $d = e^{-1} = 61^{-1} \bmod 352$
EEA:

$$\begin{array}{rcrcrcr} 352 & / & 61 & = & 5 & R & 47 \\ 61 & / & 47 & = & 1 & R & 14 \\ 47 & / & 14 & = & 3 & R & 5 \\ 14 & / & 5 & = & 2 & R & 4 \\ 5 & / & 4 & = & 1 & R & 1 \end{array}$$

$$\begin{array}{rcccc} 1 & = & 5 - 1 \cdot 4 & = & 5 - 1 \cdot (14 - 2 \cdot 5) \\ & = & -1 \cdot 14 + 3 \cdot 5 & = & -1 \cdot 14 + 3 \cdot (47 - 3 \cdot 14) \\ & = & 3 \cdot 47 - 10 \cdot 14 & = & 3 \cdot 47 - 10 \cdot (61 - 47) \\ & = & -10 \cdot 61 + 13 \cdot 47 & = & -10 \cdot 61 + 13 \cdot (352 - 5 \cdot 61) \\ & = & 13 \cdot 352 - 75 \cdot 61 & & \end{array}$$

$\Rightarrow d = -75 = 277 \bmod 352$

(b) Sei $\mathcal{A}$ ein PPT-Algorithmus, der bei Eingabe eines RSA-Modulus $N = P \cdot Q$ mit Wahrscheinlichkeit $\epsilon_{\mathcal{A}}$ den Wert $\varphi(N)$ ausgibt. Wir konstruieren einen PPT-Algorithmus $\mathcal{B}$, der bei Eingabe N mit Wahrscheinlichkeit $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}}$ die Faktoren P und Q ausgibt.

$\mathcal{B}$ ruft bei Eingabe N den Algorithmus $\mathcal{A}$ mit Eingabe N auf. Mit Wahrscheinlichkeit $\epsilon_{\mathcal{A}}$ gibt $\mathcal{A}$ den Wert $\varphi(N)$ aus.

Zunächst ein paar Beobachtungen: Offensichtlich gilt $\varphi(N) = (P-1)(Q-1) = PQ - P - Q + 1 = N - P - Q + 1$, also gilt

$$Q = N - \varphi(N) - P + 1. \tag{1}$$

Da $N = PQ$ ist, können wir Gleichung (1) einsetzen und erhalten $N = P(N - \varphi(N) - P + 1) = -P^2 + (N - \varphi(N) + 1)P$. Somit erhalten wir

$$-P^2 + (N - \varphi(N) + 1)P - N = 0. \tag{2}$$

Gleichung (2) ist eine quadratische Gleichung in P, die per Konstruktion eine natürliche Lösung hat. Diese Lösung kann z.B. durch

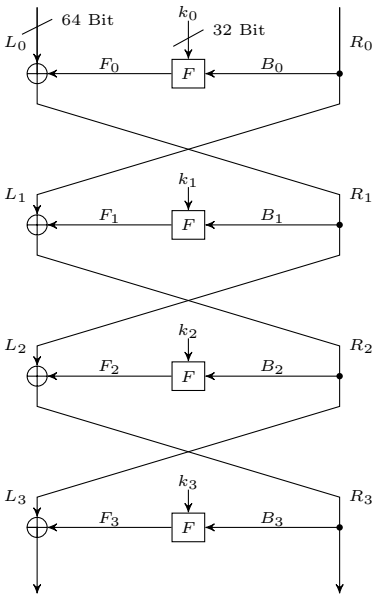
$$P = \frac{N - \varphi(N) + 1 \pm \sqrt{(N - \varphi(N) + 1)^2 + 4N}}{2} \tag{3}$$

in polynomieller Zeit berechnet werden.

Falls $\mathcal{A}$ den Wert $\varphi(N)$ liefert, kann $\mathcal{B}$ also P wie in Gleichung (3) mit Wahrscheinlichkeit 1 in polynomieller Zeit berechnen. Daher ist $\mathcal{B}$ mit Wahrscheinlichkeit $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}}$ erfolgreich, was nach Voraussetzung nicht-vernachlässigbar ist.

Der Fall, dass $\mathcal{A}$ nicht erfolgreich ist, muss nicht weiter betrachtet werden. In diesem Fall wird Gleichung (2) keine natürliche Lösung haben. Das Verhalten von $\mathcal{B}$ in diesem Fall muss nicht weiter spezifiziert werden.

Aufgabe 2. $(4 + 3 + 1 + 2$ Punkte)
Gegeben sei folgendes Feistel-Netzwerk:



Die Länge der Nachrichten und Chiffrate beträgt jeweils 128 Bit und jeder Rundenschlüssel hat eine Länge von 32 Bit. Über die verwendete F-Funktion ist bekannt, dass die Parität des 4. Eingabebits zusammen mit den 16. und 42. Ausgabebits immer ungerade ist. Das heißt, dass für alle $k \in \{0,1\}^{32}$ und $b, f \in \{0,1\}^{64}$ mit $F(k,b) = f$ folgende Gleichung erfüllt ist:

$$b[4] \oplus f[16] \oplus f[42] = 1.$$

(a) Erweitern Sie diese Beobachtung auf einen Zusammenhang zwischen L_0, R_0, L_3 und R_3. Geben Sie Ihren Rechenweg an. Diesen Zusammenhang nennt man auch eine 3-Runden-Charakteristik.

Hinweis: Sie dürfen die Kurzschreibweise aus der Übung verwenden.

(b) Die 3-Runden-Charakteristik aus Aufgabenteil (a) ermöglicht es einen Angriff auf den Rundenschlüssel k_3 durchzuführen. Beschreiben Sie, wie dieser Angriff funktioniert.

Sie müssen das weitere Vorgehen um die Rundenschlüssel k_0, k_1, k_2 zu berechnen nicht beschreiben.

(c) Welche Informationen müssen einem Angreifer für den Angriff aus Aufgabenteil (b) zur Verfügung stehen?

(d) Funktioniert der Angriff aus Aufgabenteil (b) auch dann, wenn die 3-Runden-Charakteristik zusätzlich von mindestens einem Bit der Schlüssel k_0, k_1, k_2 abhängt? Die 3-Runden-Charakteristik gibt nun also einen Zusammenhang zwischen Bits aus L_0, R_0, L_3, R_3, k_0, k_1 und k_2 an. Begründen Sie Ihre Antwort.

Lösungsvorschlag zu Aufgabe 2.

(a) Die 3-Runden-Charakteristik lautet:

$$L_0[16,42] \oplus R_0[4] \oplus L_3[4] \oplus R_3[16,42] = 0$$

Rechenweg (ausführlich):

$$\begin{aligned}
1 &= B_0[4] \oplus F_0[16,42] \\
&= R_0[4] \oplus L_0[16,42] \oplus R_1[16,42] \\
&= R_0[4] \oplus L_0[16,42] \oplus L_2[16,42] \\
&= R_0[4] \oplus L_0[16,42] \oplus F_2[16,42] \oplus R_3[16,42] \\
&= R_0[4] \oplus L_0[16,42] \oplus B_2[4] \oplus 1 \oplus R_3[16,42] \\
&= R_0[4] \oplus L_0[16,42] \oplus R_2[4] \oplus 1 \oplus R_3[16,42] \\
&= R_0[4] \oplus L_0[16,42] \oplus L_3[4] \oplus 1 \oplus R_3[16,42]
\end{aligned}$$

(Zur Schreibweise: mit $N[i_1, i_2, \ldots, i_l]$ für einen Bitstring $N = N_0 \,\|\, N_1 \,\|\, \ldots \,\|\, N_{n-1} \in \{0,1\}^n$ und Indices $i_j \in \{0, \ldots, n-1\}$ ist die Parität der Bits an den Stellen i_j von N gemeint, also $N[i_1, i_2, \ldots, i_l] := \bigoplus_{j=1}^{l} N_{i_j}$.)

(b) Vollständige Suche über alle möglichen Rundenschlüssel der letzten Runde.
Für jeden solchen Rundenschlüssel k_3 entschlüssle alle gegebenen Chiffrate um je eine Runde und erhalte Zwischenwerte L_3, R_3. Prüfe, ob Charakteristik für jede jeweils zugehörige Nachricht zusammen mit den Zwischenwerten L_3, R_3 erfüllt ist. Falls dies bei einem Klartext-Chiffrat-Paar nicht gilt, kann Rundenschlüssel verworfen werden.

(c) Klartext-Chiffrat-Paare müssen bekannt sein (Known-Plaintext-Attack)

(d) ja
Grund: Charakteristik hat die Form $L_0[A] \oplus R_0[B] \oplus L_3[C] \oplus R_3[D] = k_0[E] \oplus k_1[F] \oplus k_2[G] \oplus c$, wobei c eine Konstante ist. Die rechte Seite der Gleichung ist konstant. Hat die linke Seite der Gleichung also bei einem festen Kandidaten für k_3 nicht bei allen Klartext-Chiffrat-Paaren den gleichen Wert, so kann dieser Kandidat für k_3 verworfen werden.

Aufgabe 3. $(10(= 2 + 1 + 7) + 2 + 2$ Punkte)

(a) Betrachten wir nun kryptographische Hashfunktionen.

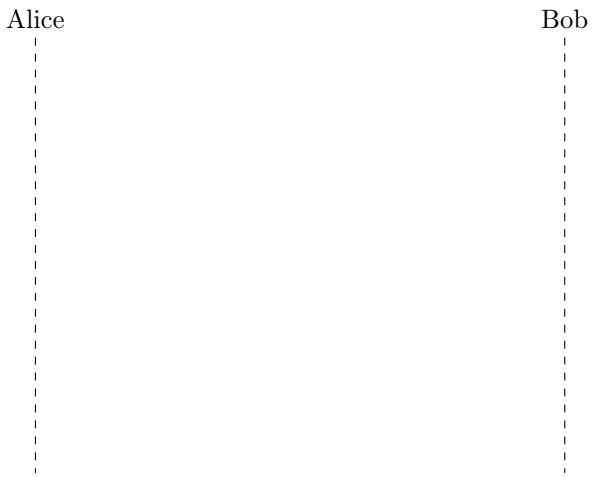
(i) Sei $H\colon \{0,1\}^* \to \{0,1\}^k$ eine über k parametrisierte effizient berechenbare Abbildung. Definieren Sie formal, wann H eine Einwegfunktion bezüglich der Urbildverteilungen $\{U_k\}_{k\in\mathbb{N}}$ ist.

(ii) Sei $H\colon \{0,1\}^* \to \{0,1\}^k$ eine kryptographische Hashfunktion. Welchen Aufwand hinsichtlich Rechenzeit und Speicherbedarf erfordert der Birthday-Angriff aus der Vorlesung gegen die Hashfunktion H?

(iii) Sei $H\colon \{0,1\}^* \to \{0,1\}^k$ eine kollisionsresistente Hashfunktion. Konstruieren Sie aus H eine kollisionsresistente Hashfunktion $\widetilde{H}\colon \{0,1\}^* \to \{0,1\}^{\widetilde{k}}$, die keine Einwegfunktion bezüglich der Urbildverteilungen $\{U_k\}_{k\in\mathbb{N}}$ ist, wobei mit U_k die Gleichverteilung auf $\{0,1\}^k$ gemeint ist. Die Ausgabelänge $\widetilde{k}$ von $\widetilde{H}$ soll dabei höchstens $2k$ sein.

Beweisen Sie, dass Ihre Konstruktion für $\widetilde{H}$ kollisionsresistent aber keine Einwegfunktion bezüglich der Urbildverteilungen $\{U_k\}_{k\in\mathbb{N}}$ ist.

(b) In der Vorlesung wurde das Diffie-Hellman-Schlüsselaustauschverfahren eingeführt. Dazu sei G eine Gruppe mit primer Ordnung p und Erzeuger g. Ergänzen Sie den Ablauf dieses Verfahrens zwischen den zwei Parteien Alice und Bob in der Zeichnung unten. Instantiieren Sie das Verfahren in der Gruppe G.



(c) Betrachten wir nun die in der Vorlesung vorgestellte Sicherheitslücke Cross-Site Scripting (XSS). Beschreiben Sie diese Sicherheitslücke im Allgemeinen.

Lösungsvorschlag zu Aufgabe 3.

(a) (i) Die Abbildung H ist eine Einwegfunktion bezüglich der Urbildverteilungen $\{U_k\}_{k\in\mathbb{N}}$, falls für alle PPT-Algorithmen $\mathcal{A}$ die Wahrscheinlichkeit

$$\Pr[X \leftarrow U_k,\, X' \leftarrow \mathcal{A}(1^k, H(X))\colon H(X) = H(X')]$$

vernachlässigbar (in k) ist.

(ii) Angriff (nicht gefragt):

- Wähle $2^{\frac{k}{2}}$ viele zufällige Urbilder x_i aus $\{0,1\}^{2k}$ und berechne jeweils den Hashwert $y_i := H(x_i)$
- speichere diese Tupel in einer Liste und sortiere die Liste nach der zweiten Komponente
- suche nach identischen y_i

(Erfolgswahrscheinlichkeit: $> \frac{1}{11}$)
Laufzeit: $O(k \cdot 2^{\frac{k}{2}})$ (Grund für Faktor k: Sortieren)
Speicherbedarf: $O(k \cdot 2^{\frac{k}{2}})$ Bits ($3k$ Bits pro Tabellenzeile)

(iii)

$$\widetilde{H}\colon \{0,1\}^* \to \{0,1\}^{k+1}$$
$$x \mapsto \begin{cases} 0 \,\|\, x & \text{falls } x \in \{0,1\}^k \\ 1 \,\|\, H(x) & \text{sonst} \end{cases}$$

Kollisionsresistenz:
Angenommen es existiert ein PPT-Angreifer $\mathcal{A}$ auf die Kollisionsresistenz von $\widetilde{H}$, der eine nicht-vernachlässigbare Erfolgswahrscheinlichkeit $\epsilon_{\mathcal{A}}$ hat. Wir konstruieren daraus einen PPT-Angreifer $\mathcal{B}$ auf die Kollisionsresistenz von H. $\mathcal{B}$ soll eine Kollision von H finden. $\mathcal{B}$ ruft $\mathcal{A}$ auf, der mit Wahrscheinlichkeit ϵ_A eine Kollision von $\widetilde{H}$ ausgibt, also zwei Nachrichten $X \neq X'$, sodass $\widetilde{H}(X) = \widetilde{H}(X')$ gilt. $\mathcal{B}$ gibt dann X und X' aus.

Falls X und X' eine Kollision für die Hashfunktion $\widetilde{H}$ sind, sind sie auch eine Kollision für die Hashfunktion H, denn:
Die beiden Nachrichten X und X' müssen außerhalb von $\{0,1\}^k$ liegen. Denn angenommen die Nachricht X liegt in $\{0,1\}^k$, dann hat der zugehörige Hashwert $\widetilde{H}(X)$ die Form $0 \,\|\, X$, weswegen auch X' in $\{0,1\}^k$ liegen muss und $X = X'$ gelten muss (denn es gilt ja $\widetilde{H}(X) = \widetilde{H}(X')$). Dies ist ein Widerspruch zu $X \neq X'$. Daher gilt $X, X' \notin \{0,1\}^k$. Die Hashwerte zu den Nachrichten X und X' haben also die Form $\widetilde{H}(X) = 1 \,\|\, H(X)$ beziehungsweise $\widetilde{H}(X') = 1 \,\|\, H(X)$. Aus $\widetilde{H}(X) = \widetilde{H}(X')$ folgt also $H(X) = H(X')$. Da $X \neq X'$ gilt, ist X, X' auch eine Kollision für die Hashfunktion H.

Einwegeigenschaft:
Sei U_k die Gleichverteilung auf der Menge $\{0,1\}^k$. Konstruiere einen PPT-Angreifer $\mathcal{A}$ auf die Einwegeigenschaft bezüglich U_k: $\mathcal{A}$ erhält als Eingabe $\widetilde{H}(X)$ für ein X aus U_k (d.h. ein gleichverteilt zufälliges X aus $\{0,1\}^k$). Da $X \in \{0,1\}^k$, ist $\widetilde{H}(X) = 0 \,\|\, X$. $\mathcal{A}$ erhält also als Eingabe $0 \,\|\, X$, entfernt die führende 0 und gibt X aus.

(b) Alice: $a \leftarrow \mathbb{Z}_p$, $A := g^a$ (kein modulo!), Alice $\xrightarrow{A}$ Bob
Bob: $b \leftarrow \mathbb{Z}_p$, $B := g^b$ (kein modulo!), Bob $\xrightarrow{B}$ Alice, Schlüssel: $K := A^b$ (kein modulo!)
Alice: Schlüssel: $K := B^a$ (kein modulo!)

(c)
- Webseite erlaubt Nutzern das Veröffentlichen von Nachrichten (Forum, Pinnwand, ...)
- Angreifer veröffentlicht auf diesem Weg ausführbaren Javascript-Code
- Benutzer, die Webseite aufrufen, empfangen diesen Code; Browser führt Code aus
- Dieser Code läuft bei Opfern in vertrauenswürdigem Kontext und hat daher Zugriff auf Cookies und Funktionen von (vertrauenswürdiger) Webseite

Aufgabe 4. (1 + 3 + 2 + 5 Punkte)

Betrachten wir nun Verfahren zur symmetrischen Authentifikation von Nachrichten.

(a) Sei $H\colon \{0,1\}^* \to \{0,1\}^k$ eine beliebige kollisionsresistente Hashfunktion. Geben Sie an, wie beim HMAC-Verfahren aus der Vorlesung der Message Authentication Code für eine Nachricht $M \in \{0,1\}^*$ berechnet wird.

(b) Sei $F\colon \{0,1\}^{2k} \to \{0,1\}^k$ eine kollisionsresistente Kompressionsfunktion. Verwenden Sie die Merkle-Damgård-Konstruktion um aus F eine kollisionsresistente Hashfunktion H zu konstruieren. Definieren Sie H formal, eine Zeichnung ist nicht ausreichend.

(c) Sei H die Hashfunktion aus Aufgabenteil (b). Wir betrachten das MAC-Verfahren, bei dem der Message Authentication Code σ zu einer Nachricht $M \in \{0,1\}^*$ durch $\sigma := H(K \,\|\, M)$ berechnet wird, wobei $K \in \{0,1\}^k$ ein zufällig gewählter Bitstring ist, der nur dem Sender und dem Empfänger der Nachricht bekannt ist.

Ist dieses MAC-Verfahren EUF-CMA-sicher? Begründen Sie Ihre Antwort kurz.

(d) Sei $E\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ eine Blockchiffre.

Wir betrachten folgendes MAC-Verfahren (Sig, Ver) für Nachrichten aus $\{0,1\}^*$, deren Länge ein Vielfaches von k ist:

$\mathsf{Sig}(K, M)$

$l := \dfrac{|M|}{k}$

$M =: M_1 \,\|\, M_2 \,\|\, \ldots \,\|\, M_l$ (wobei $M_i \in \{0,1\}^k$)

$\gamma_0 := 0^k$

$\gamma_i := E(K, M_i \oplus \gamma_{i-1})$ für $1 \le i \le l$

return $\sigma := \gamma_l$

$\mathsf{Ver}(K, M, \sigma)$

$l := \dfrac{|M|}{k}$

$M =: M_1 \,\|\, M_2 \,\|\, \ldots \,\|\, M_l$ (wobei $M_i \in \{0,1\}^k$)

$\gamma_0 := 0^k$

$\gamma_i := E(K, M_i \oplus \gamma_{i-1})$ für $1 \le i \le l$

if $\gamma_l == \sigma$ **then**

  return 1

else

  return 0

Beweisen Sie, dass dieses MAC-Verfahren nicht EUF-CMA-sicher ist, indem Sie einen EUF-CMA-Angreifer konstruieren. Zeigen Sie, dass Ihr EUF-CMA-Angreifer polynomielle Zeitkomplexität und eine nicht-vernachlässigbare Erfolgswahrscheinlichkeit im EUF-CMA-Spiel hat.

Lösungsvorschlag zu Aufgabe 4.

(a)

$$H\Big((K \oplus opad) \,\|\, H\big((K \oplus ipad) \,\|\, M\big)\Big)$$

Für K, $ipad$, $opad \in \{0,1\}^k$

(b) Eingabe: $M \in \{0,1\}^*$ mit $|M| \le 2^k$

- Setze $B := \lceil \frac{L}{k} \rceil$.
- Füge Padding hinzu:
$$M \| 0^{k - L \bmod k} = M_1 \| \ldots \| M_B$$
- Nun insgesamt B k-Bit-Blöcke $M_1, \ldots, M_B$.
- Setze $M_{B+1} := L$, wobei L mit exakt k Bits kodiert wird (z.B. binär mit führenden Nullen).
- Setze $Z_0 := 0^k$.
- Für $i = 1, \ldots, B+1$ berechne $Z_i := F(Z_{i-1} \| M_i)$.
- Gib Z_{B+1} aus.

(c) Bei Merkle-Damgård kann aus $H(K \,\|\, M)$ ohne Kenntnis von K der Hashwert zu einer Nachricht der Form $K \,\|\, M \,\|\, X$ berechnet werden. X hat dabei die Form $l \,\|\, X'$ für beliebiges $X' \in \{0,1\}^*$, wobei $l := |K \,\|\, M|$ mit genau k Bits kodiert ist.

Der Angreifer (nicht notwendig für volle Punktzahl) könnte z.B. so aussehen:

- Angreifer $\mathcal{A}$ erhält Zugriff auf ein Orakel, das ihm MACs für beliebige Nachrichten berechnet.
- $\mathcal{A}$ fragt eine beliebige Nachricht M an und erhält $H(K \,\|\, M)$.
- $\mathcal{A}$ wählt beliebige Nachricht X' und berechnet $\sigma^* := H(K \,\|\, M \,\|\, l \,\|\, X')$ (wobei $l := |K \,\|\, M|$ mit genau k Bits kodiert) wie oben und gibt $m^* := K \,\|\, M \,\|\, l \,\|\, X'$ und $\sigma*$ aus.

(d) Nicht EUF-CMA-sicher! Konstruiere Angreifer $\mathcal{A}$.

- $\mathcal{A}$ hat Zugriff auf MAC-Orakel
- wähle Nachrichten $m \in \{0,1\}^{k \cdot l}$, $m' \in \{0,1\}^{k \cdot l'}$ beliebig (für bel. l, l')
- frage m und m' beim Orakel an, erhalte jeweils σ und σ'
- sei $m' =: m'_1 \,\|\, m'_2 \,\|\, \ldots \,\|\, m'_{l'}$
- $m'' := m \,\|\, m'_1 \oplus \sigma \,\|\, m'_2 \,\|\, m'_3 \,\|\, \ldots \,\|\, m'_{l'}$
- gib m'' und σ' aus

σ' ist eine gültige Signatur für m'', denn $C_l = \sigma$, $C_{l+1} = E(K, C_l \oplus \sigma \oplus m'_1) = E(K, m'_1)$, $\Rightarrow$ $C_{l+l'} = \sigma'$. $\mathcal{A}$ läuft in Polynomialzeit und hat Erfolgswahrscheinlichkeit 1.

Alternative Lösung:
Idee: frage Nachrichten der Länge $1 \cdot k$ beim Orakel an. Dies liefert für eine Nachricht m als Antwort $E(K, m \oplus 0^k) = E(K, m)$. Daher kann ein Angreifer auf diese Weise für beliebige Nachrichten $m \in \{0,1\}^{l \cdot k}$ Fälschungen produzieren:

- $\mathcal{A}$ wählt $m \in \{0,1\}^{l \cdot k}$ beliebig, $m =: m_1 \,\|\, m_2 \,\|\, \ldots \,\|\, m_l$
- berechne $C_0 := 0^k$, $C_i := E(K, m_i \oplus C_{i-1}) = \mathcal{O}(m_i \oplus C_{i-1})$
- gib m und C_l aus

σ ist offensichtlich eine gültige Signatur für m. $\mathcal{A}$ läuft in Polynomialzeit und hat Erfolgswahrscheinlichkeit 1.

Aufgabe 5. (3 + 6 + 5 Punkte)

Sei $G = (V, E)$ ein ungerichteter Graph. Der Einfachheit halber sei $V = \{1, \ldots, n\}$ für $n \in \mathbb{N}$. Ein Hamiltonkreis in G ist ein Kreis in G, der jeden Knoten von G genau einmal besucht. (Ein Pfad kann hier als eine Folge von Knoten $(v_1, v_2, \ldots, v_l)$ aufgefasst werden, sodass für alle $1 \leq i \leq l-1$ die Kanten $\{v_i, v_{i+1}\}$ in E sind. Ein Kreis ist ein Pfad bei dem der Anfangs- und der Endknoten identisch sind.)

Wir betrachten ein Public-Key-Identifikationsschema (Gen, Prove, Ver), bei dem ein „Prover" Prove einen „Verifier" Ver davon überzeugen will, dass er einen Hamiltonkreis in einem vorgegebenen Graphen G kennt, ohne dabei Informationen über diesen Hamiltonkreis zu verraten.

- Die Parametergenerierung $\mathsf{Gen}(1^k)$ erzeugt einen Graphen $G = (V, E)$ zusammen mit einem zugehörigen Hamiltonkreis $C \in V^{|V|+1}$. Schließlich wird der öffentliche Schlüssel $pk := G$ und der geheime Schlüssel $sk := (G, C)$ ausgegeben. (Dieser Schritt wird von einer vertrauenswürdigen Instanz zu Beginn einmalig ausgeführt.)

Prove erhält als Eingabe den geheimen Schlüssel sk, Ver erhält als Eingabe den öffentlichen Schlüssel pk. Der Protokollablauf ist wie folgt definiert:

1.) Prove wählt eine zufällige Permutation $\varphi\colon V \to V$ der Knoten von G (also eine zufällige Umbenennung der Knoten von G), berechnet $H := \varphi(G)$ und committet sich auf die Kanten von H.[1] (Mit $\varphi(G)$ ist der Graph $\varphi(G) := (V, E')$ gemeint mit $E' := \{\{\varphi(u), \varphi(v)\} \mid \{u, v\} \in E\}$.)

2.) Ver wählt ein Bit $b \leftarrow \{0, 1\}$ zufällig gleichverteilt und sendet es an Prove.

3.) Ist $b = 0$, so sendet Prove den verwendeten Isomorphismus φ an Ver und öffnet die Commitments aus Schritt 1.).[2]
Ist $b = 1$, so sendet Prove den Hamiltonkreis $\varphi(C)$ in H an Ver und öffnet die Commitments für jene Kanten, die in $\varphi(C)$ verwendet werden.[3] (Für einen Pfad $C = (v_1, v_2, \ldots, v_l)$ ist mit $\varphi(C)$ der Pfad $(\varphi(v_1), \varphi(v_2), \ldots, \varphi(v_l))$ gemeint.)

4.) Ver akzeptiert genau dann, wenn die jeweiligen Commitments korrekt geöffnet wurden und eine der folgenden Bedingungen gilt:
   - Es gilt $b = 0$ und $\varphi(G) = H$.
   - Es gilt $b = 1$ und der Pfad $\varphi(C)$ ist ein Hamiltonkreis in H (um dies zu überprüfen, verwendet Ver die geöffneten Commitments auf die benötigten Kanten von H).

(a) Begründen Sie kurz, wieso das obige Protokoll korrekt ist.

(b) In diesem Aufgabenteil nehmen wir an, dass bereits im Voraus bekannt ist, welches Bit b der Verifier in Schritt 2.) wählen wird. Dann können wir für $b \in \{0, 1\}$ jeweils einen PPT-Angreifer $\mathcal{A}_b$ in der Rolle des Provers konstruieren, der nur den öffentlichen Schlüssel $pk = G$ als Eingabe erhält und den Verifier trotzdem zum Akzeptieren bringt.

Geben Sie das Vorgehen von $\mathcal{A}_0$ und $\mathcal{A}_1$ in Schritt 1.) und Schritt 3.) an. Begründen Sie insbesondere, wieso Ihr $\mathcal{A}_b$ den Verifier Ver zum Akzeptieren bringt, wenn dieser in Schritt 2.) das Bit b ausgibt. (Sie müssen nicht begründen, wieso Ihr $\mathcal{A}_b$ polynomielle Laufzeit hat.)

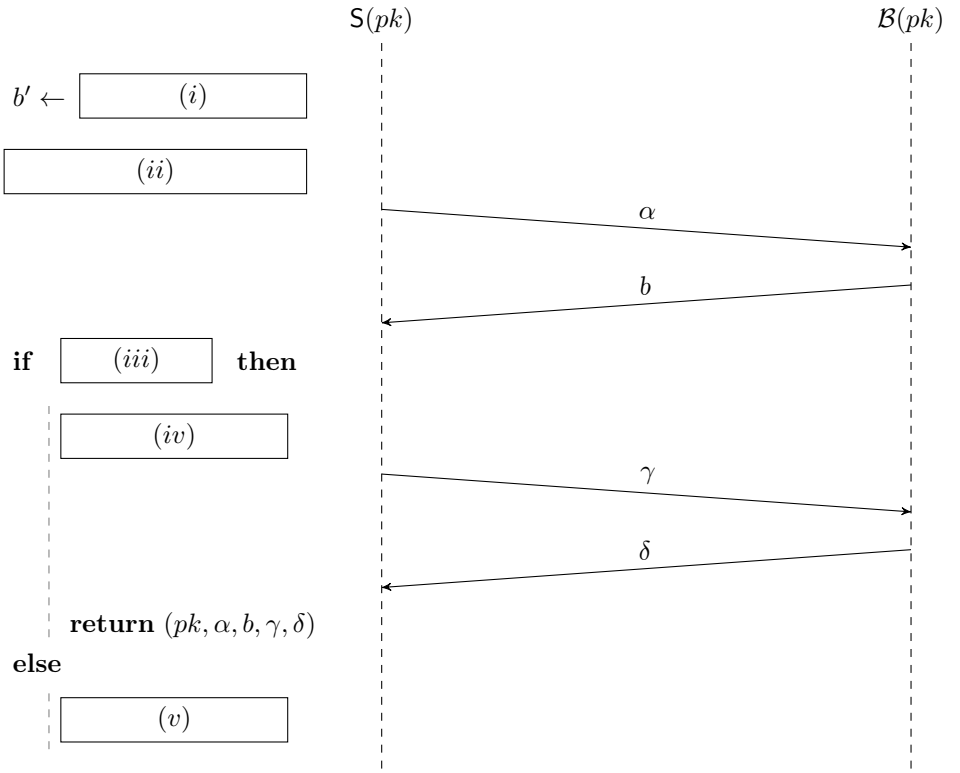
Hinweis: Nehmen Sie dazu an, dass ein PPT-Algorithmus $\mathtt{GenHC}(n, m)$ gegeben ist, der einen zufälligen Graphen $\widetilde{G}$ mit n Knoten und m Kanten zusammen mit einem zugehörigen Hamiltonkreis $\widetilde{C}$ ausgibt.

(c) Sei nun $\mathcal{B}$ ein PPT-Angreifer in der Rolle des Verifiers. Geben Sie einen Simulator S an, der Transkripte ausgibt, die von Transkripten eines ehrlichen Provers (der sk kennt) mit $\mathcal{B}$ in der Rolle des Verifiers nicht unterscheidbar sind. Ergänzen Sie dazu die Beschriftungen in folgendem Diagramm, indem Sie die unten stehende Tabelle ausfüllen. Achten Sie darauf, dass die erwartete Laufzeit von S höchstens polynomiell ist.

[1] Sei Com ein Commitment-Verfahren. Prove committet sich auf die Kanten von H, indem er die Menge $\{\mathsf{Com}(e; R_e) \mid e \text{ Kante von } H,\ R_e \text{ zufällig}\}$ an Ver sendet.

[2] Prove öffnet die Commitments auf die Kanten von H, indem er $\{(e, R_e) \mid e \text{ Kante von } H\}$ an Ver sendet

[3] Prove öffnet die Commitments auf die Kanten von $\varphi(C)$, indem er $\{(e, R_e) \mid e \text{ Kante von } \varphi(C)\}$ an Ver sendet.



(i)	
(ii)	
(iii)	
(iv)	
(v)	

Hinweis: Verwenden Sie Aufgabenteil (b). Sie dürfen Aufgabenteil (b) auch dann verwenden, wenn Sie dafür keine Lösung haben.

Lösungsvorschlag zu Aufgabe 5.

(a) Wenn Prove einen Hamiltonkreis C in G kennt, kann er beide möglichen Anfragen von Ver aus Schritt 2.) in Schritt 3.) so beantworten, dass Ver schließlich in Schritt 4.) akzeptiert.
Genauer:

- im Falle $b = 0$ öffnet Prove die Commitments auf alle Kanten und sendet die gewählte Permutation φ an Ver
- im Falle $b = 1$ berechnet Prove den Hamiltonkreis $\varphi(C)$ und sendet ihn an Ver, zudem öffnet er die Commitments auf alle in $\varphi(C)$ vorkommenden Kanten

(b) **Fall 1:** $\mathcal{A}_0$ geht davon aus, dass Ver in Schritt 2.) das Bit $b = 0$ ausgibt:
$\mathcal{A}_0$ wählt in Schritt 1.) eine zufällige Permutation φ der Knoten von G und committet sich auf die

Kanten von $H := \varphi(G)$. $\mathcal{A}_0$ geht also zu Beginn genau so vor wie ein ehrlicher Prover. Wählt Ver in Schritt 2.) also wirklich $b = 0$, so antwortet $\mathcal{A}_0$ in Schritt 3.) wie der ehrliche Verifier indem er die Commitments auf die Kanten von H aufdeckt und die Permutation φ bekannt gibt.
In Schritt 4.) prüft Ver, ob die Commitments auf die Kanten von H korrekt geöffnet wurden und, ob $\varphi(G) = H$ gilt. Diese Überprüfungen sind beide erfolgreich, da $\mathcal{A}_0$ den Graph H als $\varphi(G)$ berechnet hat (und die Commitments auf die Kanten von H ehrlich erzeugt hat).

Fall 2: $\mathcal{A}_1$ geht davon aus, dass Ver in Schritt 2.) das Bit $b = 1$ ausgibt:
In Schritt 1.) generiert $\mathcal{A}_1$ einen zufälligen von G unabhängigen Graphen G' der gleichen Größe zusammen mit einem zugehörigem Hamiltonkreis C' mit $\mathtt{GenHC}(n, m)$ und committet sich auf die Kanten von G'. Wählt Ver in Schritt 2.) also wirklich $b = 1$, so antwortet $\mathcal{A}_1$ in Schritt 3.) indem er den Hamiltonkreis C' von G' ausgibt und die Commitments auf alle Kanten von C' öffnet. In Schritt 4.) prüft Ver, ob die geöffneten Commitments korrekt geöffnet wurden und prüft mit Hilfe der nun bekannten Kanten von G', ob C' ein Hamiltonkreis in G' ist. Diese Überprüfungen sind erfolgreich, da $\mathcal{A}_1$ den Graphen G' so wählt, dass er einen Hamiltonkreis C' in G' kennt (und die Commitments auf die Kanten von G' ehrlich erzeugt werden).

(c)

(i)	$\leftarrow \{0,1\}$ zufällig
(ii)	$:= \mathcal{A}_{b'}(pk)$
(iii)	$b == b'$
(iv)	$:= \mathcal{A}_{b'}(b)$
(v)	Rewinding, gehe zurück zum Anfang (also wähle auch b' neu)